(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 697 655 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
27.03.2002 Bulletin 2002/13

(51) Int Cl.[7]: **G06F 9/44**, G06F 9/46

(21) Application number: **95305614.0**

(22) Date of filing: **11.08.1995**

(54) System for managing external applications and files

System zur Verwaltung von externen Anwendungen und Dateien

Systéme pour gérer des applications et des fichiers externes

(84) Designated Contracting States:
DE FR GB

(30) Priority: **19.08.1994 US 293373**

(43) Date of publication of application:
21.02.1996 Bulletin 1996/08

(73) Proprietor: **CANON KABUSHIKI KAISHA**
Tokyo (JP)

(72) Inventor: **Bibayan, Farzad,**
c/o Canon Kabushiki Kaisha
Ohta-ku, Tokyo (JP)

(74) Representative:
Beresford, Keith Denis Lewis et al
BERESFORD & Co.
2-5 Warwick Court,
High Holborn
London WC1R 5DH (GB)

(56) References cited:
EP-A- 0 304 072

- **IBM TECHNICAL DISCLOSURE BULLETIN, vol. 32, no. 9B, 1 February 1990, NEW YORK, US, page 80/81 XP000082241 "CONVERSATIONALLY DEPENDENT MULTIPLE TASK TERMINATION"**

Note: Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

EP 0 697 655 B1

BEST AVAILABLE COPY

Description

[0001] The present invention is a computer-implemented management system which works in conjunction with a client application program so as to manage external application programs and their associated files. More particularly, the management system of the present invention provides a system for a client application to launch, track and otherwise manage an external application and its associated file.

[0002] It has recently become possible to create a computer document which contains data from a variety of different application programs. Such a document is called a "compound document". Compound documents are created by a first application program, usually called a "client application", and contain one or more data objects from other application programs. A compound document might, for example, be created by a word processing application program. In this case, the compound document contains not only text data native to the word processing application program, but also contains data objects from other application programs such as a bit map data object from a graphics display program, a spreadsheet data object from a spreadsheet program, or a sound data object from a sound recorder application program.

[0003] Compound documents are an important development in computer systems because they eliminate the constraint that data in a document must be that one single type of data supported by the client application. With compound documents, it is now possible to create a word processing document that includes not only text from the word processing application but which also includes numbers from a spreadsheet application, images from a graphics application, and sounds from a recorder application.

[0004] Figures 1(a) through 1(c) illustrate how a compound document might appear to a user of a client application program, here a client application program which allows a user to send and to receive multimedia messages which may include attachments from external application programs. A suitable client application program for sending and receiving multimedia messages is described in detail in application Serial No. 07/808,757, filed December 17, 1991 (copy filed).

[0005] In Figure 1(a), a compound document 1 is shown as it might be displayed to a user on display screen 2 of computer monitor 3. Compound document 1 is a message that is scheduled to be sent to a remote recipient. The compound document includes a text portion 4 which is data native to the client message manager application program. In addition, compound document 1 includes attachments signified by icons 5, 6 and 7 which are data objects from external application programs. Specifically, text attachment 5 is a simple text file created by an external text editor application such as Microsoft Note Pad. Bitmap attachment 6 is a bitmap file created by an external graphics program such as Microsoft Paint Brush. Sound attachment 7 is a wave file created by an external sound application program such as Microsoft Sound Recorder.

[0006] By selecting one of the icons in compound document 1, such as by double clicking on the icon with a mouse, the user can cause the external application which created the data object to be launched so as to view, edit or otherwise modify the data object associated with the icon. Thus, as shown in Figure 1(b), after selecting the bitmap icon 6, the external Paint Brush application program is launched so as to display the bitmap attachment. Likewise, in Figure 1(c), the user has selected the remaining icons so as to cause associated external application programs to be launched.

[0007] Various implementations have been proposed so as to allow an external application to be launched from a client application which is displaying a compound document. For example, Microsoft has introduced object linking and embedding (hereinafter "OLE") which allows data objects from external applications to be linked or embedded in a compound document for a client application. OLE is described in detail in Chapter 9 of F. Davis, "The Windows 3.1 Bible", Peachpit Press, 1993. OLE provides a robust implementation of compound documents which allows a user to launch external applications, to switch from one application to another, and to monitor how objects in the compound document are being modified by their respective external applications. However, to achieve this implementation, specialized communication is needed between the external application and the client application. For example, the external application must be OLE-aware and must register itself into a "registration database". The client application must, in addition, be OLE-compliant, meaning that the client application must be able to communicate directly or indirectly to the external applications so as to determine whether they are present,must support the creation of complex compound documents, and must also conform to standard OLE interfaces.

[0008] Other implementations of compound documents have been proposed, but these implementations constrain how a user is able to manipulate data objects in the compound document. For example, "CC Mail" from Lotus, which is an E-mail client application, provides a compound document capability which is simpler than Microsoft's OLE but which does not provide as many features as OLE. For example, when an external application is launched from the "CC Mail" client application, a user is not permitted to access any other application program until the external application has been closed down. This limits the usefulness of compound documents. For example, when using the client "CC Mail" application program, the user would not be able to obtain the display in Figure 1(c). Rather, the user would be constrained to the display shown in Figure 1(b) and would not be able to access any other program, including the client application program, until the Note Pad external application program had been closed down. Par-

ticularly in a windowing operating system such as Microsoft Windows, in which a user expects to be able to access any application program at any time, this is a significant limitation.

[0009] Accordingly, there is a need to provide a managing system for managing external applications associated with compound documents in which the management system provides a user with flexibility like that provided in an OLE-based implementation while at the same time does not require specialized communication between the client application program and the external application program.

[0010] The present invention as claimed addresses the aforementioned need in the art by providing a managing system which manages external applications that are launched from compound documents displayed in a client application without the need for specialized communication between the client application and the external applications.

[0011] In one aspect, the management system of the present invention includes an external application manager and an external file manager, both of which are instantiated by a client application program which supports compound documents. The external application manager determines which external application to launch based on characteristics in a file name provided from the client application, such as the DOS extension to the file name, launches the external application with the file, returns a pointer for the external application to the client application, periodically polls for activity of the external application, and outputs the file name to the external file manager. The external file manager monitors the file to determine if the file has been modified by the external application. Modification information is provided via the application manager back to the client application.

[0012] Because the external application manager simply monitors the external application (by polling), and the external file manager likewise only monitors the external file, there is no need for specialized communication between the client application and the external application. Moreover, because the external application is launched using a windowing operating system, the user is free to switch between any of the applications, including the client application. Thus, for example, if the external application is no longer in focus when the user reselects the external data object from the client application display, the client application simply refers to pointers returned from the application manager to determine whether there is an existing application program corresponding to the external object. If there is, then the client application requests application manager to put the external application in focus, and the application manager in turn asks the windowing operating system to activate the window of the external application so as to put the external application in focus. If there is no pointer, then the client application simply requests the application manager and file manager to launch a suitable external application.

[0013] Likewise, when an external application is closed down, the application manager which has been monitoring the activity of external applications advises the client application of this fact. The client application then decides based on the monitoring activities of the file manager whether it is necessary to re-import a file that has been modified by an external application.

[0014] In like manner, when the client application is closed down or iconized, the client application can easily determine whether there are any external applications by referring to the pointers returned by the application manager. If the client application is being closed down, then it can prompt the user to close down all external applications prior to closing down; as the external applications are closed, the client application can re-import any modified files as described previously. On the other hand, if the client application is being iconized, then the client application can sweep all external applications down into a single icon for the client application.

[0015] Accordingly, the invention as described above provides a full featured and robust implementation of compound documents which does not require a client application to communicate with external applications.

[0016] This brief summary has been provided so that the nature of the invention may be understood quickly. A more complete understanding of the invention can be obtained by reference to the following detailed description of the preferred embodiment thereof in connection with the attached drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0017] Figures 1(a), 1(b) and 1(c) illustrate how a compound document appears to a user of a client application program.

[0018] Figure 2 is a view for showing the outward appearance of a representative embodiment of the present invention.

[0019] Figure 3 is a detailed block diagram showing the internal construction of computing equipment shown in Figure 2.

[0020] Figure 4 is a block diagram showing the structure of the managing system of the present invention.

[0021] Figure 5 is a flow diagram illustrating operation of the managing system shown in Figure 4.

[0022] Figure 6 is an illustrative view of a compound document.

[0023] Figure 7 is a detailed flow diagram showing polling by the application tracker.

[0024] Figure 8 is a flow diagram showing the procedure for closing down or iconizing a client application.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0025] Figure 2 is a view showing the outward appearance of a representative embodiment of the present in-

vention. Shown in Figure 2 is computing equipment 20 such as an IBM PC or PC-compatible computer having a windowing operating system such as a Microsoft Windows operating system. Computing equipment 20 is provided with a display monitor 23 having a display screen 22 on which computing equipment 20 displays images to the user. Computing equipment 20 is further provided with a floppy disk drive 24 with which removable floppy disk media may be read or written, fixed disk drive 25 for storing data files and application program files, a keyboard 26 for permitting input of text data and manipulation of objects displayed on display screen 22, a pointing device 27 such as a mouse or the like which is also provided to permit manipulation of objects on display screen 22, and a combined speaker/microphone 29. A conventional printer 30 as well as connections to a network 31 or to an ordinary voice telephone line 32 for sending and receiving voice and/or facsimile messages are also provided.

[0026] In accordance with operator instructions, and under control of the windowing operating system, stored application programs such as client application programs and external application programs are selectively activated to process and manipulate data.

[0027] Figure 3 is a detailed block diagram showing the internal construction of computing equipment 20. As shown in Figure 3, computing equipment 20 includes a central processing unit (CPU) 40 such as programmable microprocessor interfaced to a computer bus 41. Also interfaced to computer bus 41 is speaker/microphone interface 42, display interface 44, network interface 45, fax/modem/telephone interface 46, printer interface 47, and floppy disk drive interface 49.

[0028] Main memory 51 such as random access memory (RAM) interfaces to computer bus 41 so as to provide CPU 40 with access to memory storage. In particular, when executing stored application program instruction sequences such as those associated with application programs stored on disk 25, CPU 40 loads those instruction sequences from disk 25 (or other storage media such as media accessed via network 31) into main memory 51 and executes those stored program instruction sequences out of main memory 51.

[0029] ROM (read only memory) 52 is provided for storing invariant instruction sequences such as start-up instruction sequences or basic input/output operating system (BIOS) sequences for operation of keyboard 26.

[0030] As shown in Figure 3, and as previously mentioned, fixed disk 25 stores program instruction sequences for the windowing operating system and for various application programs such as a message manager application program, a graphics application program, a sound application program, and the like. In addition, stored on fixed disk 25 are compound document files together with other data files such as data files associated with the application programs. Fixed disk 25 also stores the management system of the present invention which, as shown in Figure 3, is comprised by two components: an external application manager and an external file manager. Both components are instantiated by a client application that requires managing of external applications and their associated files. The structure of the external application and file managers is described in connection with Figure 4, and its operation is described in the flow diagrams in Figures 5, 7 and 8.

[0031] More particularly, Figure 4 shows client application 55 with instantiations of application manager 60 and file manager 70. Application manager 60 includes application pool manager 61, application tracker 62, application launcher 64, application finder 65, and application browser 66. Figure 4 shows application manager 60 as containing three application entry objects, namely objects 67, 68 and 69. As will be described more fully below, one such object is provided for each external application, such as applications 87, 88 and 89. It will therefore be understood that the provision of three application entry items is illustrative only, and that typically upon instantiation of application manager 60 there will not be any application entry objects until client application 55 requests application manager 60 to launch such an external application.

[0032] File manager 70 includes file pool manager 71, and file tracker 72. In addition, Figure 4 shows file manager 70 as including three file entry objects, namely objects 77, 78 and 79, one each for each of external files such as files 97, 98 and 99. As before with respect to application entry objects 67, 68 and 69, one file entry object is provided for each external file. Thus, although Figure 4 shows three such objects, upon instantiation of file manager 70 there will ordinarily not be any file entry objects.

[0033] The functional capabilities of each object shown in Figure 4 will now be explained.

[0034] Application pool manager 61 is a holder for all application entry objects and has the capability of creating, adding, removing and managing those objects. Application pool manager 61 communicates with client application 55 by receiving requests from client application 55 to create entry objects and by sending to client application 55 pointers for application entry objects. Application pool manager 61 also communicates with file pool manager 71 by providing file pool manager 71 with file names and by receiving file pointers from file pool manager 71.

[0035] Application tracker 62 is responsible for checking for existence of all external applications and for synchronizing itself with the status of those external applications. As described more fully below, application tracker 62 operates on a periodic polling basis, namely by sending polling requests to application finder 65 and to file tracker 72. In situations where application finder 65 advises application tracker 62 that a particular external application no longer exists, application tracker 62 advises application pool manager 61 of the new application status. In that case, the application pool manager

61 removes the appropriate application entry object and, in turn, updates client application 55's list of pointers. File tracker 72 responds to polling requests from application tracker 62 by advising application tracker 62 of the status of external files, specifically whether such external files have been modified or not. In the event that external files have been modified, application tracker 62 advises application pool manager 61 of that fact and application pool manager 61, in turn, updates client application 55.

[0036] Application launcher 64 is used to launch an external application. Application launcher 64 communicates with client application 55 through receipt of a data object. Ordinarily, the data object is a file name received from client application 55 in response to user selection of a data object in a compound document. In that case, application launcher 64 determines whether there is an external application associated to the file name. If there is an external application associated with the file name application launcher 64 launches that application. To find associated applications, application launcher 64 issues requests to the windowing operating system for associated applications. If the windowing operating system cannot identify the application, application launcher 64 tries to identify an associated application through other means, such as searching through initialization (.INI) files. If an application is successfully launched, application launcher 64 advises client application 55 of a successful launch. Otherwise, application launcher 64 advises client application 55 that no application was launched.

[0037] Application browser 66 is invoked by client application 55 in the event that an external application cannot be associated to a data object provided to application launcher 64. If invoked, application browser 66 provides the user with a display of available application programs and permits the user to select one. If an application is selected, application browser 66 provides the selected application to client application 55 which, in turn, provides it to application launcher 64.

[0038] Application entry objects, such as objects 67, 68 and 69, contain all related information for applications such as the task handle for the application, the instance handle for the application, the window handle of the application, and the owner of the application (here, an item within client application 55). As mentioned above, one application entry object is provided for each external application.

[0039] Application finder 65 is responsible for finding previously-launched external applications and assuring that any external applications so found are actually present. In particular, because of timing delays and system overhead, it is possible to incorrectly conclude that an application is present when in fact it no longer exists. To assure that an external application is actually present, application finder 65 obtains the application window handle and the application task handle. Application finder 65 then matches the application instance handle in the application entry object against the application instance handle obtained from the window handle, and matches the application task handle in the application entry object against the application task handle obtained from the window handle. If both match, application finder 65 concludes that an application which the windowing operating system indicates is present is actually present, and advises application tracker 62 of that in response to polling requests.

[0040] File pool manager 71 holds all file entry objects and has the capability of creating, adding, removing and managing those objects. File pool manager 71 communicates with application pool manager 61 by receiving file names from application pool manager 61. In response to receipt of a file name, file pool manager 71 opens a file entry object, and returns a file pointer to the application pool manager.

[0041] File tracker 72 checks for existence of all files in the file pool and determines when a file has been modified. When it detects modification of a file, file tracker 72 advises application tracker 62 of that fact in response to a polling request.

[0042] File entry objects, such as objects 77, 78 and 79, contain all related information for a file such as the file name, a time stamp and possible persistent storage identifiers. As mentioned above, file pool manager 71 creates only one file entry object for each file.

[0043] By virtue of the above-described structure of application manager 60 and file manager 70, it will be appreciated that a full-featured external application and file management system has been provided which does not need to communicate with any external application or any external file. Rather, to provide full management of external applications and files, the application manager and file manager need only check pointers and handles provided by the windowing operating system.

[0044] Figure 5 is a flow diagram illustrating operation of the Figure 4 managing system. The steps performed in Figure 5 are executed by CPU 30 in accordance with stored program instruction sequences stored on fixed disk 25 and executed out of random access memory 51.

[0045] In step S501, the user activates the client application which, as part of its initialization routine, instantiates the application manager 60 and file manager 70. In step S502, application tracker 62 commences polling of application finder 65 and file tracker 72 to determine the status of any existing external applications as well as whether or not files associated with those external applications have been modified. Polling is described in fuller detail in connection with Figure 7; for present purposes, it is sufficient to note that polling occurs on a periodic basis which is set to between three and ten seconds.

[0046] In step S503, the user utilizes client application 55 to create a compound document, such as the compound document shown in Figure 6. More particularly, in the representative embodiment of the invention described here, the client application is a message man-

ager application program. To build a compound document, the user selects the message builder option button 101 from tool bar 100, which causes the message manager application to display a message builder screen 102. After entering data in the native message manager application mode, the user is permitted to attach documents created by external applications, thereby creating a compound document. In particular, by selecting attach button 105, screen 106 is displayed to allow a user to embed a document or form a link to the document. In the case of embedding a document, a copy of the document is physically created within the compound document, whereas in the case of a linked document, only a link to the external document is stored. The option of linking or embedding is provided by radio button 103a and 103b. After selecting whether to link or to embed the external document, a compound document such as that shown in display area 104 is created.

[0047] In step S504, the client application displays the compound document and allows user selection of any external object (or data item) from the compound document. In step S505, in response to user selection of a data item, client application 55 checks if there is a valid pointer, previously supplied from application pool manager 61, for the entry object for this item (S506). If there is a valid pointer, then the user has previously activated this data item and there is a corresponding external application which is present. Accordingly flow branches to step S507 in which client application 55 merely passes the window handle to the windowing operation system which, in turn, puts the external application into focus for use by the user. Flow then advances to step S519 in which the windowing operating system displays a window for the selected external application.

[0048] Presumably, however, if this is the first time that the item has been selected by the user, then there is no valid pointer for an entry object for this item. Accordingly, flow continues to step S508 in which the client application creates a temporary file for use by the external application. In more detail, in a case where the item is a linked item, client application 55 simply uses the file designated by the link. If, on the other hand, the selected item is an embedded item, then client application 55 creates a temporary file and stores the embedded data from the compound document into the temporary file. When creating the temporary file, client application 55 maintains the same DOS extension for the file name. As will be seen below, the DOS extension is used as a key to associate particular external application programs with the temporary file. For example, a .BMP extension is presumed to be a bitmap file and an external graphics application such as Microsoft Paint Brush which handles bitmap files, is associated with a .BMP file. Likewise, a .TXT file is presumed to be a text file and a text editor such as Microsoft Note Pad is associated with such a file; and a .WAV file is assumed to be a wave file and a sound editor such as Microsoft Sound Recorder is associated with such a file. Similar associations are well known to those skilled in the art.

[0049] In step S509, client application 55 invokes application launcher 64 to determine whether an external application can be associated with the temporary file (or, for a linked item, the linked file). Application launcher 64 inspects the DOS extension of the file name and determines whether an external application can be associated with that DOS extension. In making this determination, application launcher 64 refers to the windowing operating system and, in the case where the windowing operating system cannot return an external application, also refers to initialization files such as .INI files which store such associations. If an association is not found (step S510), then flow advances to step S511 in which application launcher 64 returns an error code to client application 55. Client application 55 then invokes application browser 66 (step S512) which allows a user to make a manual selection of an external application (step S513). The manual selection is returned by application browser 66 to client application 55 which, in turn, returns the manually-selected application to application launcher 64.

[0050] In the case where application launcher 64 is able to associate an external application file to the DOS extension, or in the case of manual designation by the user in step S513, application launcher 64 launches the associated external application. For example in the case of a .BMP file, application launcher 64 may launch a Paint Brush application as seen at 87. Similarly, in the case of a .TXT file, application launcher 64 launches a Note Pad application 88; or in the case of a .WAV file, application launcher 64 launches sound recorder external application 89. Application launcher then informs client application 55 that an external application has successfully been launched (step S515).

[0051] Upon learning of successful launch of an external application, client application 55 invokes application pool manager 64 to create an application entry object corresponding to the launched application. Application pool manager 61 responds by creating an application entry object such as object 67, 68 or 69, the application entry object being responsible to store information concerning the application. As mentioned previously, one application entry object is created for each existing external application (or in the case where two different files in the same compound document have the same DOS extension, for each separate instance of the external application).

[0052] Application pool manager 61 further invokes file pool manager 71 to create a corresponding file entry object (step S517). In like manner to the application pool manger, the file pool manager 71 creates a file entry object, such as object 77, 78 or 79, to track the status of the temporary file with which the newly-launched external application is operating.

[0053] Flow then advances to step S518 in which application pool manger returns a pointer to the client application. The pointer is a pointer for the application entry

object and the file entry object. The client application 55 is able to use this pointer for future operations, such as in step S506 where the client application determines whether there is a valid pointer signifying that an external application is present for a selected item, or such as described below in connection with closing down or iconizing of client application 55. Flow then advances to step S519 in which the windowing operating system displays a window for the selected external application.

[0054] Thereafter, the user is free to use the newly-launched external application to create, edit or otherwise modify the data item from the compound document. In addition, the user is free to switch focus to other windows or to activate new windows. When the user wishes to return to the window corresponding to the external application, he can do so through normal windowing protocol or he may reselect the data item from a compound document displayed by client application 55. In that case flow proceeds as described above in connection with step S506 in which client application 55 determines whether there is a valid pointer for the entry object corresponding to the selected item and in which, in response to a positive identification of such a valid pointer, client application 55 simply passes the window handle to the windowing operating system which executes focus onto the external application.

[0055] Figure 7 is a detailed flow diagram showing polling by application tracker 62.

[0056] In step S701, application tracker 62 invokes file tracker 72 which, via any existing file entry objects, determines whether a file has been modified. If file tracker 72 determines from a file entry object that an external file has been modified (step S702) then flow branches to step S703 in which file tracker 72 advises application tracker 62 that a particular file has been modified. That information is passed from application tracker 62 to application pool manager 61 which, in turn, advises client application 55 that a particular file corresponding to one file entry object has been modified.

[0057] In either event, flow advances to step S704 in which application tracker 62 invokes application finder 65 to determine whether each external application corresponding to an application entry object is still present. Using the above-described procedure of matching windows handles and instance handles to task handles, application finder 65 checks to ensure that all external applications corresponding to application entry objects are still present. So long as the tested handle values match (step S706), then application finder 65 determines that each external application is still present and returns that information to application tracker 62 (step S707). Then, after a predetermined period (such as three to ten seconds), application tracker 62 repeats its polling process (step S708).

[0058] On the other hand, if in step S706 application finder 65 determines that there is no match for the tested handle values, then application finder 65 concludes that an external application has been closed down. Flow then advances to step S709 in which close down procedures are carried out. Particularly, as shown in step S710, application tracker 62 advises application pool manager to remove an application entry object corresponding to the closed down external application. In step S711, the closed down application entry object updates application finder 65 with its new closed down status and then closes itself down. Then in step S712, application pool manager 61 invokes file pool manager 71 to remove its corresponding file entry object. In response, in step S713, the file entry object (or objects) determines if the temporary file created in step S508 was modified by the external application. If no modification had taken place, then client application 55 simply destroys the temporary file. On the other hand, if modifications have taken place, then client application 55 reimports the data in the temporary file into its internal compound document. In addition, client application 55 updates any parameters affected by the re-imported file, such as parameters giving an indication to the user as to size and/or date of most recent modification of the file. In step S714 application pool manager 61 advises client application 55 that the external application has closed down.

[0059] Figure 8 is a flow diagram showing the procedure in a case where the user has selected close down or iconization of client application 55. More particularly, as is generally known, it is possible for the user to request the windowing operating system to close down client application 55, for example, by double clicking on control button 107 shown in Figure 6. Likewise, it is possible for a user to request the windowing operating system to iconize the client application, for example by clicking on minimize button 108. When close down is requested, the windowing operating system exits client application 55 and frees memory previously used by that application. If iconization is selected, the windowing operating system removes all windows associated with client application 55 from display screen 22 and replaces all those windows with an icon representing the client application. The user may re-invoke all windows associated with client application 55 by double clicking the application icon.

[0060] The flow process shown in Figure 8 ensures that upon close down or iconization of client application 55, corresponding external applications are closed down in an orderly procedure or swept into the icon for client application 55. More particularly, as shown in step S801, upon user selection of close down or iconization, client application 55 determines whether there are any external applications present. Client application 55 makes this determination by reference to its list of pointers for entry objects which was provided by application pool manager. If no external applications exist (step S802), flow branches to step S803 in which client application 55 is simply closed down or iconized, as requested by the user.

[0061] If, on the other hand, there are any existing ex-

ternal applications, then flow advances to step S804 in which, if a close down operation has been requested, flow advances to step S805 where client application 55 requests the user to close down the external applications. A dialog box may be provided to warn the user of the existence of external applications. As each external application is closed down, the close down procedures commencing at step S709 are followed, after which flow advances to step S806 in which client application 55 is closed down.

[0062] If step S804 determined that an iconize operation had been requested, then flow advances to step S807 in which client application 55 provides the windowing operating system with the window handle to each external application and requests the windowing operating system to de-display that window. After all external applications have had their windows de-displayed, flow advances to step S808 in which client application 55 is reduced to an icon. In response to a user request to restore client application 55 (step S809) such as by double clicking on the iconic representation of client application 55, the windowing operating system re-displays the client application 55 (step S810). After the client application 55 has been redisplayed, it, in turn, provides the windowing operating system with pointers for all external applications. Using those windows pointers, the windowing operating system re-displays the external applications.

Claims

1. A computer implemented system for managing an external application (87, 88, 89) in response to user selection of an item in a compound document displayed from a client application (55), said computer implemented system **characterised by**:

   an application manager (60) which receives a file name for an external file (97, 98, 99) corresponding to the item selected from the compound document and which, based on the file name, associates an external application (87, 88, 89) to the external file, launches the external application, repetitively polls for information regarding the external application, and outputs the file name, wherein the information polled for at least includes information concerning existence of the external application (87, 88, 89); and

   a file manager (70), responsive to outputting of the file name from the application manager (60), which monitors the external file to determine if the external file (97, 98, 99) has been modified by the external application and which returns to the application manager (60) a pointer for the external file (97, 98, 99);

   wherein the application manager (60) returns a pointer for the external application (87, 88, 89) and a pointer for the external file (97, 98, 99) to the client application (55).

2. A computer implemented system according to claim 1, wherein the client application (55) responds to user selection of an item in the compound document by copying the selected item to a temporary file and by sending the temporary file name to the application manager (60).

3. A computer implemented system according to claim 2, wherein said application manager (60) includes:

   (a) an application tracker (62) which repetitively outputs polling requests to check for existence of previously-launched external applications,
   (b) an application finder (65) responsive to polling requests from the application tracker (62) which returns to the application tracker (62) information on whether an external application (87, 88, 89) is present,
   (c) an application pool manager (61) which creates, adds and removes application entry objects, the application pool manager (61) also being for receiving the external file name from the client application (55) and outputting it, and for returning a pointer for the external file (97, 98, 99) and the external application (87, 88, 89) to the client application, and
   (d) an application launcher (64) which associates an external application to the external file name output by said client application (55) and which launches the external application (87, 88, 89).

4. A computer implemented system according to claim 3, wherein said file manager (70) includes:

   (i) a file tracker (72) responsive to polling requests from the application tracker (62) to check for modification of the external file (97, 98, 99) created by the client application returning information on file modifications to the application tracker (62), and
   (ii) a file pool manager (71) for creating, adding and removing a file entry object corresponding to the external file (97, 98, 99), the file pool manager (71) being responsive to output of the external file name from the application pool manager (61) by creating a file entry object.

5. A computer implemented system according to any of claims 1 to 4, wherein in response to receipt of the pointers from the application manager (60), the client application (55) instructs a windowing operating system to display a window for the external ap-

plication.

6. A computer implemented system according to any of claims 1 to 4, wherein the client application (55) responds to user selection of an item in the compound document by determining if a valid pointer for the selected item has been received from the application manager (60), and wherein in a case where a valid pointer exists the client application (55) instructs a windowing operating system to display a window for the external application, and wherein in a case where a valid pointer does not exist the client application (55) copies the selected item to a temporary file, sends the temporary file name to the application manager (60), and in response to receipt of a pointer from the application manager (60), instructs the windowing operating system to display a window for the external application (87, 88, 89).

7. A computer implemented system according to any of claims 1 to 4, wherein in response to close down of an external application, said application manager (60) invalidates the pointer sent to the client application (55).

8. A computer implemented system according to claim 7, wherein in response to close down of an external application, the external file manager determines whether the external file (97, 98, 99) has been modified, and in a case where the external file has been modified reimports data from the external file (97, 98, 99).

9. A computer implemented system according to any of claims 1 to 4, wherein in response to close down of the client application (55), the client application (55) inspects valid pointers received from the application manager (60) to determine whether any external applications are still present, and before closing down requests close-down of still-present external applications.

10. A computer implemented system according to any of claims 1 to 4, wherein in response to iconization of the client application (55), the client application (55) inspects valid pointers received from the application manager (60) to determine whether any external applications (87, 88, 89) are still existing and instructs a windowing operating system to de-display any still-existing external applications before iconization.

11. A computer implemented system according to claim 10, wherein in response to restoration of an iconized client application, the client application (55) inspects valid pointers received from said application manager (60) and instructs the windowing operating system to re-display any still-present external applications.

12. A computer implemented system according to any of claims 1 to 4, wherein information polled for includes a window handle for the external application (87, 88, 89) and a task handle for the external application (87, 88, 89), and wherein said application manager (60) determines whether the external application (87, 88, 89) is present by matching the window handle and the task handle against corresponding window and task handles in an entry object for the external application (87, 88, 89).

13. A computer implemented method for managing an external application in response to user selection of an item in a compound document displayed from a client application (55), said method comprising the steps of:

instantiating (S501) an application manager (60) which repetitively commences polling operations to determine whether previously-launched external applications are present and whether associated external files have been modified (S502);
instantiating (S501) a file manager (70) responsive to polling requests from the application manager (60) to determine whether an external file has been modified and to inform the application manager of same;
displaying (S504) a compound document from the client application (55); and
responding to user selection (S505) of an item in the compound document by creating (S508) an external file corresponding to the selected item, outputting a file name corresponding to the selected item to the application manager (60), associating (S509) an external application to the file name, launching (S514) the external application, and returning (S518), to the client application (55), a pointer for the external file and the external application.

14. A computer implemented method according to claim 13, wherein said step of responding to user selection of an item comprises the following steps performed by the client application (55); copying the selected item to a temporary file (S508), and sending the temporary file name to the application manager (60).

15. A computer implemented method according to claim 14, wherein said application manager includes:

(a) an application tracker (62) which repetitively outputs polling requests to check for existence

of previously-launched external applications,
(b) an application finder (65) responsive to polling requests from the application tracker (62) which returns to the application tracker (62) information on whether an external application is present,
(c) an application pool manager (61) which creates, adds and removes application entry objects, the application pool manager (61) also for receiving the external file name from the client application (55) and outputting it, and for returning a pointer for the external file (97, 98, 99) and the external application to the client application, and
(d) an application launcher (64) which associates an external application to the external file name output by said client application (55) and which launches the external application.

16. A computer implemented method according to claim 15, wherein said file manager (70) includes:

(i) a file tracker (72) responsive to polling requests from the application tracker to check for modification of the external file created by the client application returning information on file modifications to the application tracker, and
(ii) a file pool manager (71) for creating, adding and removing a file entry object corresponding to the external file, the file pool manager (71) being responsive to output of the external file name from the application pool manager by creating a file entry object.

17. A method according to any of claims 13 to 16, further comprising the step of responding (518) to receipt of the pointer for the external file and the external application by instructing a windowing operating system to display a window for the external application (S519).

18. A method according to any of claims 13 to 16, wherein said steps of responding to user selection of an item in the compound document includes the steps of determining if a valid pointer for the selected item has been received from the application manager (S506), instructing (507) the windowing operating system to display a window for the external application in the case where a valid pointer exists, and sending (S509) an external file name to the application manager (60) in a case where a valid pointer does not exist.

19. A method according to any of claims 13 to 16, further comprising the step of responding to close down of the external application by invalidating the pointer sent to the client application (S714).

20. A method according to claim 19, further comprising the step of re-importing (S713) external file data to the compound document in a case where the external file has been modified.

21. A method according to any of claims 13 to 16, further comprising the step of responding to close down of the client application by inspecting pointers (S801) received from the application manager (60) to determine whether any external applications are still present, and requesting close-down of still-present external applications (S805) before closing down the client application (55).

22. A method according to any of claims 13 to 16, further comprising the steps of responding to iconization of the client application (55) by inspecting (S801) the pointers received from the application manager (60) to determine whether any external applications are still present and instructing the windowing operating system to de-display (S807) any still-present external applications before iconizing.

23. A method according to claim 22, further comprising the step of responding to restoration of the iconized client application by inspecting pointers received from the application manager and instructing the windowing operating system to re-display (S811) any still-present external applications.

24. A computer implemented method according to any of claims 13 to 16, wherein information polled for includes a window handle for the external application and a task handle for the external application, and wherein said application manager (60) determines whether the external application is present by matching the window handle and the task handle against corresponding window and task handles in an entry object for the external application.

25. An apparatus for managing an external application in response to user selection of an item in a compound document displayed from a client application, said apparatus comprising:

a display (22) for displaying compound documents and windows corresponding to the client application and external applications;
a memory (25) for storing program steps corresponding to a windowing operating system which responds to instructions to display the client application and external applications on said display (22), and for storing program instruction sequences corresponding to the client application; and
a processor (40) for executing the stored program instruction sequences in said memory;

characterized by said stored program instruction sequences including steps to perform any of the methods of claims 13 to 24.

26. A computer software product carrying machine readable instructions effective to cause a processing apparatus to carry out a method according to any one of claims 13 to 24.

Patentansprüche

1. Mit einem Computer realisiertes System zum Verwalten einer externen Anwendung (87, 88, 89) als Reaktion auf eine Auswahl eines Elements in einem von einer Client-Anwendung (55) angezeigten Container-Dokument durch einen Anwender, **gekennzeichnet durch:**

   eine Anwendungsverwaltungseinrichtung (60), die einen Dateinamen für eine externe Datei (97, 98, 99) entsprechend dem aus dem Container-Dokument ausgewählten Element empfängt und die auf der Grundlage des Dateinamens eine externe Anwendung (87, 88, 89) mit der externen Datei verknüpft, die externe Anwendung startet, wiederholt zyklisch die externe Anwendung betreffenden Informationen abfragt und den Dateinamen ausgibt, wobei die zyklisch abgefragten Informationen zumindest das Vorhandensein der externen Anwendung (87, 88, 89) betreffende Informationen umfassen; und
   eine auf die Ausgabe des Dateinamens von der Anwendungsverwaltungseinrichtung (60) reagierende Dateiverwaltungseinrichtung (70), die die externe Datei zur Bestimmung, ob die externe Datei (97, 98, 99) **durch** die externe Anwendung verändert worden ist, überwacht, und die einen Zeiger für die externe Datei (97, 98, 99) an die Anwendungsverwaltungseinrichtung (60) zurückgibt;

   wobei die Anwendungsverwaltungseinrichtung (60) einen Zeiger für die externe Anwendung (87, 88, 89) und einen Zeiger für die externe Datei (97, 98, 99) an die Client-Anwendung (55) zurückgibt.

2. Mit einem Computer realisiertes System nach Anspruch 1, wobei die Client-Anwendung (55) auf die Auswahl eines Elements in dem Container-Dokument durch den Benutzer reagiert, indem sie das ausgewählte Element in eine temporäre Datei kopiert und den Namen der temporären Datei an die Anwendungsverwaltungseinrichtung (60) sendet.

3. Mit einem Computer realisiertes System nach Anspruch 2, wobei die Anwendungsverwaltungseinrichtung (60) umfasst:

   (a) eine Anwendungsverfolgungseinheit (62), die wiederholt zyklisch abfragende Anforderungen zur Überprüfung auf das Vorhandensein vorher gestarteter externer Anwendungen ausgibt,
   (b) eine auf zyklisch abfragende Anforderungen von der Anwendungsverfolgungseinheit (62) reagierende Anwendungssucheinheit (65), die Informationen darüber an die Anwendungsverfolgungseinheit (62) zurückgibt, ob eine externe Anwendung (87, 88, 89) vorhanden ist,
   (c) eine Anwendungspoolverwaltungseinrichtung (61), die Anwendungseintragsobjekte erzeugt, hinzufügt und entfernt, wobei die Anwendungspoolverwaltungseinrichtung (61) auch dazu dient, den Namen der externen Datei von der Client-Anwendung (55) zu empfangen und ihn auszugeben, und dazu dient, einen Zeiger für die externe Datei (97, 98, 99) und die externe Anwendung (87, 88, 89) an die Client-Anwendung zurückzugeben, und
   (d) eine Anwendungsstarteinheit (64), die eine externe Anwendung mit dem von der Client-Anwendung (55) ausgegebenen Namen der externen Datei verknüpft und die die externe Anwendung (87, 88, 89) startet.

4. Mit einem Computer realisiertes System nach Anspruch 3, wobei die Dateiverwaltungseinrichtung (70) umfasst:

   (i) eine Dateiverfolgungseinheit (72), die auf zyklisch abfragende Anforderungen zur Überprüfung auf eine Änderung der von der Client-Anwendung erzeugten externen Datei (97, 98, 99) von der Anwendungsverfolgungseinheit (62) hin reagiert und Informationen über Dateiänderungen an die Anwendungsverfolgungseinheit (62) zurückgibt, und
   (ii) eine Dateipoolverwaltungseinrichtung (71) zum Erzeugen, Hinzufügen und Entfernen eines der externen Datei (97, 98, 99) entsprechenden Dateieintragsobjekts, wobei die Dateipoolverwaltungseinrichtung (71) auf die Ausgabe des Namens der externen Datei von der Anwendungspoolverwaltungseinrichtung (61) reagiert, indem sie ein Dateieintragsobjekt erzeugt.

5. Mit einem Computer realisiertes System nach einem der Ansprüche 1 bis 4, wobei die Client-Anwendung (55) als Reaktion auf den Empfang der Zeiger von der Anwendungsverwaltungseinrichtung (60) ein Fensterbetriebssystem dazu anweist,

ein Fenster für die externe Anwendung anzuzeigen.

6. Mit einem Computer realisiertes System nach einem der Ansprüche 1 bis 4, wobei die Client-Anwendung (55) auf die Auswahl eines Elements in dem Container-Dokument durch den Benutzer reagiert, indem sie es bestimmt, ob ein gültiger Zeiger für das ausgewählte Element von der Anwendungsverwaltungseinrichtung (60) empfangen worden ist, und wobei in einem Fall, in dem ein gültiger Zeiger vorhanden ist, die Client-Anwendung (55) ein Fensterbetriebssystem dazu anweist, ein Fenster für die externe Anwendung anzuzeigen, und wobei in einem Fall, in dem kein gültiger Zeiger vorhanden ist, die Client-Anwendung (55) das ausgewählte Element in eine temporäre Datei kopiert, den Namen der temporären Datei an die Anwendungsverwaltungseinrichtung (60) sendet und als Reaktion auf den Empfang eines Zeigers von der Anwendungsverwaltungseinrichtung (60) das Fensterbetriebssystem dazu anweist, ein Fenster für die externe Anwendung (87, 88, 89) anzuzeigen.

7. Mit einem Computer realisiertes System nach einem der Ansprüche 1 bis 4, wobei die Anwendungsverwaltungseinrichtung (60) als Reaktion auf ein Schließen einer externen Anwendung den an die Client-Anwendung (55) gesendeten Zeiger für ungültig erklärt.

8. Mit einem Computer realisiertes System nach dem Anspruch 7, wobei es die externe Dateiverwaltungseinrichtung als Reaktion auf das Schließen einer externen Anwendung bestimmt, ob die externe Datei (97, 98, 99) verändert worden ist, und in einem Fall, in dem die externe Datei verändert worden ist, Daten aus der externen Datei (97, 98, 99) neu importiert.

9. Mit einem Computer realisiertes System nach einem der Ansprüche 1 bis 4, wobei die Client-Anwendung (55) als Reaktion auf ein Schließen der Client-Anwendung (55) von der Anwendungsverwaltungseinrichtung (60) empfangene gültige Zeiger untersucht, um es zu bestimmen, ob noch externe Anwendungen vorhanden sind, und vor dem eigenen Schließen ein Schließen der noch vorhandenen externen Anwendungen anfordert.

10. Mit einem Computer realisiertes System nach einem der Ansprüche 1 bis 4, wobei die Client-Anwendung (55) als Reaktion auf eine Verkleinerung der Client-Anwendung (55) auf Icon-Größe von der Anwendungsverwaltungseinrichtung (60) empfangene gültige Zeiger untersucht, um es zu bestimmen, ob noch externe Anwendungen (87, 88, 89) vorhanden sind, und ein Fensterbetriebssystem dazu anweist, vor der Verkleinerung auf Icon-Größe noch vorhandene externe Anwendungen von der Anzeige zu löschen.

11. Mit einem Computer realisiertes System nach dem Anspruch 10, wobei die Client-Anwendung (55) als Reaktion auf eine Wiederherstellung einer auf Icon-Größe verkleinerten Client-Anwendung von der Anwendungsverwaltungseinrichtung (60) empfangene gültige Zeiger untersucht und das Fensterbetriebssystem dazu anweist, noch vorhandene externe Anwendungen wieder anzuzeigen.

12. Mit einem Computer realisiertes System nach einem der Ansprüche 1 bis 4, wobei zyklisch abgefragte Informationen ein Fenster-Handle für die externe Anwendung (87, 88, 89) und ein Task-Handle für die externe Anwendung (87, 88, 89) umfassen und wobei es die Anwendungsverwaltungseinrichtung (60) bestimmt, ob die externe Anwendung (87, 88, 89) vorhanden ist, indem das Fenster-Handle und das Task-Handle mit dem entsprechenden Fenster- und Task-Handle in einem Eintragsobjekt für die externe Anwendung (87, 88, 89) verglichen werden.

13. Mit einem Computer realisiertes Verfahren zum Verwalten einer externen Anwendung als Reaktion auf eine Auswahl eines Elements in einem von einer Client-Anwendung (55) angezeigten Container-Dokument durch einen Benutzer, mit den Schritten:

Instantiieren (S501) einer Anwendungsverwaltungseinrichtung (60), die wiederholt zyklisch abfragende Operationen beginnt, um es zu bestimmen, ob vorher gestartete externe Anwendungen vorhanden sind und ob verknüpfte externe Dateien verändert worden sind (S502);
Instantiieren (S501) einer Dateiverwaltungseinrichtung (70), die zur Bestimmung, ob eine externe Datei verändert worden ist, und zur Information der Anwendungsverwaltungseinrichtung darüber auf zyklisch abfragende Anfragen von der Anwendungsverwaltungseinrichtung (60) reagiert;
Anzeigen (S504) eines Container-Dokuments von der Client-Anwendung (55); und
Reagieren auf die Auswahl (S505) eines Elements in dem Container-Dokument durch den Benutzer durch ein Erzeugen (S508) einer dem ausgewählten Element entsprechenden externe Datei, ein Ausgeben eines dem ausgewählten Element entsprechenden Dateinamens an die Anwendungsverwaltungseinrichtung (60), ein Verknüpfen (S509) einer externen Anwendung mit dem Dateinamen, ein Starten (S514) der externen Anwendung und ein Zurückgeben (S518) eines Zeigers für die externe Datei und die externe Anwendung an die Client-Anwen-

dung (55).

14. Mit einem Computer realisiertes Verfahren nach Anspruch 13, wobei der Schritt des Reagierens auf die Auswahl eines Elements durch den Benutzer folgende von der Client-Anwendung (55) durchgeführte Schritte umfasst: Kopieren des ausgewählten Elements in eine temporäre Datei (S508) und Senden des Namens der temporären Datei an die Anwendungsverwaltungseinrichtung (60).

15. Mit einem Computer realisiertes Verfahren nach Anspruch 14, wobei die Anwendungsverwaltungseinrichtung umfasst:

(a) eine Anwendungsverfolgungseinheit (62), die wiederholt zyklisch abfragende Anforderungen zur Überprüfung auf das Vorhandensein vorher gestarteter externer Anwendungen hin ausgibt,
(b) eine auf zyklisch abfragende Anforderungen von der Anwendungsverfolgungseinheit (62) reagierende Anwendungssucheinheit (65), die Informationen darüber an die Anwendungsverfolgungseinheit (62) zurückgibt, ob eine externe Anwendung (87, 88, 89) vorhanden ist,
(c) eine Anwendungspoolverwaltungseinrichtung (61), die Anwendungseintragsobjekte erzeugt, hinzufügt und entfernt, wobei die Anwendungspoolverwaltungseinrichtung (61) auch dazu dient, den Namen der externen Datei von der Client-Anwendung zu empfangen und ihn auszugeben, und dazu dient, einen Zeiger für die externe Datei (97, 98, 99) und die externe Anwendung an die Client-Anwendung zurückzugeben, und
(d) eine Anwendungsstarteinheit (64), die eine externe Anwendung mit dem von der Client-Anwendung (55) ausgegebenen Namen der externen Datei verknüpft und die die externe Anwendung (87, 88, 89) startet

16. Mit einem Computer realisiertes Verfahren nach Anspruch 15, wobei die Dateiverwaltungseinrichtung (70) umfasst:

(i) eine Dateiverfolgungseinheit (72), die auf zyklisch abfragende Anforderungen zur Überprüfung auf eine Änderung der von der Client-Anwendung erzeugten externen Datei hin von der Anwendungsverfolgungseinheit reagiert und Informationen über Dateiänderungen an die Anwendungsverfolgungseinheit zurückgibt, und
(ii) eine Dateipoolverwaltungseinrichtung (71) zum Erzeugen, Hinzufügen und Entfernen eines der externen Datei entsprechenden Dateieintragsobjekts, wobei die Dateipoolverwaltungseinrichtung (71) auf die Ausgabe des Namens der externen Datei von der Anwendungspoolverwaltungseinrichtung (61) reagiert, indem sie ein Dateieintragsobjekt erzeugt.

17. Verfahren nach einem der Ansprüche 13 bis 16, ferner mit dem Schritt des Reagierens (518) auf den Empfang des Zeigers für die externe Datei und die externe Anwendung, indem ein Fensterbetriebssystem dazu angewiesen wird, ein Fenster für die externe Anwendung anzuzeigen (S519).

18. Verfahren nach einem der Ansprüche 13 bis 16, wobei die Schritte des Reagierens auf die Auswahl eines Elements in einem Container-Dokument durch den Benutzer, die Schritte des Bestimmens, ob ein gültiger Zeiger für das ausgewählte Element von der Anwendungsverwaltungseinheit empfangen worden ist (S506), des Anweisens (507) des Fensterbetriebssystems dazu, ein Fenster für die externe Anwendung in einem Fall anzuzeigen, in dem ein gültiger Zeiger vorhanden ist, und des Sendens (S509) eines Namens einer externen Datei an die Anwendungsverwaltungseinrichtung (60) in einem Fall, in dem kein gültiger Zeiger vorhanden ist, umfassen.

19. Verfahren nach einem der Ansprüche 13 bis 16, ferner mit dem Schritt des Reagierens auf ein Schließen der externen Anwendung, indem der an die Client-Anwendung gesendete Zeiger für ungültig erklärt wird (S714).

20. Verfahren nach Anspruch 19, ferner mit dem Schritt des Neuimportierens externer Dateidaten in das Container-Dokument in einem Fall, in dem die externe Datei verändert worden ist.

21. Verfahren nach einem der Ansprüche 13 bis 16, ferner mit dem Schritt des Reagierens auf ein Schließen der Client-Anwendung, indem die von der Anwendungsverwaltungseinrichtung (60) empfangenen Zeiger untersucht werden (S801), um es zu bestimmen, ob noch externe Anwendungen vorhanden sind, und indem ein Schließen der noch vorhandenen externen Anwendungen angefordert (S805) wird, bevor die Client-Anwendung (55) geschlossen wird.

22. Verfahren nach einem der Ansprüche 13 bis 16, ferner mit den Schritten des Reagierens auf eine Verkleinerung der Client-Anwendung (55) auf Icon-Größe durch ein Untersuchen (S801)der von der Anwendungsverwaltungseinrichtung (60) empfangenen Zeiger, um es zu bestimmen, ob noch externe Anwendungen vorhanden sind, und durch ein Anweisen des Fensterbetriebssystems dazu, vor

der Verkleinerung auf Icon-Größe noch vorhandene externe Anwendungen von der Anzeige zu löschen (S807).

23. Verfahren nach Anspruch 22, ferner mit dem Schritt des Reagierens auf eine Wiederherstellung der auf Icon-Größe verkleinerten Client-Anwendung, indem von der Anwendungsverwaltungseinrichtung empfangene Zeiger untersucht werden und das Fensterbetriebssystem dazu angewiesen wird, noch vorhandene externe Anwendungen wieder anzuzeigen (S811).

24. Mit einem Computer realisiertes Verfahren nach einem der Ansprüche 13 bis 16, wobei zyklisch abgefragte Informationen ein Fenster-Handle für die externe Anwendung und ein Task-Handle für die externe Anwendung umfassen und wobei es die Anwendungsverwaltungseinrichtung (60) bestimmt, ob die externe Anwendung vorhanden ist, indem das Fenster-Handle und das Task-Handle mit dem entsprechenden Fenster- und Task-Handle in einem Eintragsobjekt für die externe Anwendung verglichen werden.

25. Vorrichtung zum Verwalten einer externen Anwendung als Reaktion auf eine Auswahl eines Elements in einem von einer Client-Anwendung angezeigten Container-Dokument durch einen Anwender, mit:

einer Anzeige (22) zum Anzeigen von Container-Dokumenten und Fenstern entsprechend der Client-Anwendung und den externen Anwendungen;
einem Speicher (25) zum Speichern von Programmschritten, die einem Fensterbetriebssystem entsprechen, das auf Anweisungen zur Anzeige der Client-Anwendung und externer Anwendungen auf der Anzeige (22) reagiert, und zum Speichern von der Client-Anwendung entsprechenden Programmanweisungsfolgen; und
einer Verarbeitungsvorrichtung (40) zur Ausführung der gespeicherten Programmbefehlsfolgen in dem Speicher;

gekennzeichnet durch die gespeicherten Programmanweisungsfolgen, die Schritte zur Ausführung eines Verfahrens nach einem der Ansprüche 13 bis 24 umfassen.

26. Ein Computersoftwareprodukt, das maschinell lesbare Anweisungen trägt, die zur Veranlassung einer Verarbeitungsvorrichtung zur Ausführung eines Verfahrens nach einem der Ansprüche 13 bis 24 wirksam sind.

Revendications

1. Système informatique pour gérer une application externe (87, 88, 89) en réponse à une sélection par un utilisateur d'un élément dans un document composé affiché en provenance d'une application de client (55), ledit système informatique étant **caractérisé par** :

un gestionnaire d'application (60) qui reçoit un nom de fichier pour un fichier externe (97, 98, 99) correspondant à l'élément sélectionné à partir du document composé et qui, sur la base du nom de fichier, associe une application externe (87, 88, 89) au fichier externe, lance l'application externe, recueille de façon répétitive des informations concernant l'application externe, et sort le nom de fichier, lesquelles informations recueillies comprennent au moins des informations concernant l'existence de l'application externe (87, 88, 89) ; et
un gestionnaire de fichiers (70), sensible à la sortie du nom de fichier en provenance du gestionnaire d'application (60), qui contrôle le fichier externe pour déterminer si le fichier externe (97, 98, 99) a été modifié par l'application externe et qui renvoie au gestionnaire d'application (60) un pointeur pour le fichier externe (97, 98, 99) ;

dans lequel le gestionnaire d'application (60) renvoie un pointeur pour l'application externe (87, 88, 89) et un pointeur pour le fichier externe (97, 98, 99) à l'application de client (55).

2. Système informatique selon la revendication 1, dans lequel l'application de client (55) répond à la sélection par un utilisateur d'un élément dans le document composé en copiant l'élément sélectionné dans un fichier provisoire et en envoyant le nom de fichier provisoire au gestionnaire d'application (60).

3. Système informatique selon la revendication 2, dans lequel ledit gestionnaire d'application (60) comprend :

(a) un dispositif de suivi d'application (62) qui sort de façon répétitive des demandes de recueil d'informations pour vérifier l'existence d'applications externes précédemment lancées,
(b) un dispositif de recherche d'application (65) sensible aux demandes de recueil d'informations en provenance du dispositif de suivi d'application (62) qui renvoie au dispositif de suivi d'application (62) des informations pour savoir si une application externe (87, 88, 89) est présente,

(c) un gestionnaire de groupe d'applications (61) qui crée, ajoute et enlève des objets d'entrée d'application, le gestionnaire de groupe d'applications (61) servant également à recevoir le nom de fichier externe en provenance de l'application de client (55) et pour le sortir, et pour renvoyer un pointeur pour le fichier externe (97, 98, 99) et l'application externe (87, 88, 89) à l'application de client, et
(d) un dispositif de lancement d'application (64) qui associe une application externe au nom de fichier externe sorti par ladite application de client (55) et qui lance l'application externe (87, 88, 89).

4. Système informatique selon la revendication 3, dans lequel ledit gestionnaire de fichiers (70) comprend :

(i) un dispositif de suivi de fichier (72) sensible à des demandes de recueil d'informations en provenance du dispositif de suivi d'application (62) pour vérifier une modification du fichier externe (97, 98, 99) créée par l'application de client renvoyant des informations concernant des modifications de fichier au dispositif de suivi d'application (62), et
(ii) un gestionnaire de groupe de fichiers (71) pour la création, l'ajout et l'enlèvement d'un objet d'entrée de fichier correspondant au fichier externe (97, 98, 99), le gestionnaire de groupe de fichiers (71) étant sensible à une sortie du nom de fichier externe en provenance du gestionnaire de groupe d'applications (61) en créant un objet d'entrée de fichier.

5. Système informatique selon l'une quelconque des revendications 1 à 4, dans lequel en réponse à la réception des pointeurs en provenance du gestionnaire d'application (60), l'application de client (55) ordonne à un système d'exploitation à fenêtres d'afficher une fenêtre pour l'application externe.

6. Système informatique selon l'une quelconque des revendications 1 à 4, dans lequel l'application de client (55) répond à la sélection par un utilisateur d'un élément dans le document composé en déterminant si un pointeur valide pour l'élément sélectionné a été reçu en provenance du gestionnaire d'application (60), et dans lequel, dans un cas où un pointeur valide existe, l'application de client (55) ordonne à un système d'exploitation à fenêtres d'afficher une fenêtre pour l'application externe, et dans lequel, dans un cas où un pointeur valide n'existe pas, l'application de client (55) copie l'élément sélectionné dans un fichier provisoire, envoie le nom de fichier provisoire au gestionnaire d'application (60), et en réponse à la réception d'un pointeur en provenance du gestionnaire d'application (60), ordonne au système d'exploitation à fenêtres d'afficher une fenêtre pour l'application externe (87, 88, 89).

7. Système informatique selon l'une quelconque des revendications 1 à 4, dans lequel en réponse à la fermeture d'une application externe, ledit gestionnaire d'application (60) invalide le pointeur envoyé à l'application de client (55).

8. Système informatique selon la revendication 7, dans lequel en réponse à la fermeture d'une application externe, le gestionnaire de fichiers externes détermine si le fichier externe (97, 98, 99) a été modifié, et au cas où le fichier externe a été modifié, réimporte des données en provenance du fichier externe (97, 98, 99).

9. Système informatique selon l'une quelconque des revendications 1 à 4, dans lequel, en réponse à la fermeture de l'application de client (55), l'application de client (55) examine des pointeurs valides reçus en provenance du gestionnaire d'application (60) pour déterminer si de quelconques applications externes sont toujours présentes, et avant la fermeture demande la fermeture des applications externes toujours présentes.

10. Système informatique selon l'une quelconque des revendications 1 à 4, dans lequel, en réponse à la transformation en icône de l'application de client (55), l'application de client (55) examine des pointeurs valides reçus en provenance du gestionnaire d'application (60) pour déterminer si de quelconques applications externes (87, 88, 89) existent toujours et ordonne à un système d'exploitation à fenêtres de désafficher de quelconques applications externes toujours présentes avant la transformation en icône.

11. Système informatique selon la revendication 10, dans lequel en réponse à la restauration d'une application de client transformée en icône, l'application de client (55) examine des pointeurs valides reçus en provenance dudit gestionnaire d'application (60) et ordonne au système d'exploitation à fenêtres de réafficher toute application externe toujours présente.

12. Système informatique selon l'une quelconque des revendications 1 à 4, dans lequel des informations recueillies comprennent un identificateur de fenêtre pour l'application externe (87, 88, 89) et un identificateur de tâche pour l'application externe (87, 88, 89), et dans lequel ledit gestionnaire d'application (60) détermine si l'application externe (87, 88, 89) est présente en faisant correspondre l'identificateur

de fenêtre et l'identificateur de tâche à des identificateurs de fenêtre et de tâche correspondants dans un objet d'entrée pour l'application externe (87, 88, 89).

13. Procédé informatique pour gérer une application externe en réponse à la sélection par un utilisateur d'un élément dans un document composé affiché en provenance d'une application de client (55), ledit procédé comprenant les étapes consistant à :

instancier (S501) un gestionnaire d'application (60) qui commence de façon répétitive des opérations de recueil d'informations pour déterminer si des applications externes précédemment lancées sont présentes et si des fichiers externes associés ont été modifiés (S502) ;
instancier (S501) un gestionnaire de fichiers (70) sensible à des demandes de recueil d'informations en provenance du gestionnaire d'application (60) pour déterminer si un fichier externe a été modifié et pour informer le gestionnaire d'application de la même chose ;
afficher (S504) un document composé en provenance de l'application de client (55) ; et
répondre à la sélection par un utilisateur (S505) d'un élément dans le document composé en créant (S508) un fichier externe correspondant à l'élément sélectionné, en sortant un nom de fichier correspondant à l'élément sélectionné vers le gestionnaire d'application (60), en associant (S509) une application externe au nom de fichier, en lançant (S514) l'application externe, et en renvoyant (S518), à l'application de client (55), un pointeur pour le fichier externe et l'application externe.

14. Procédé informatique selon la revendication 13, dans lequel ladite étape consistant à répondre à la sélection par un utilisateur d'un élément comprend les étapes suivantes effectuées par l'application de client (55) : la copie de l'élément sélectionné dans un fichier provisoire (S508), et l'envoi du nom de fichier provisoire au gestionnaire d'application (60).

15. Procédé informatique selon la revendication 14, dans lequel ledit gestionnaire d'application comprend :

(a) un dispositif de suivi d'application (62) qui sort de façon répétitive des demandes de recueil d'informations pour vérifier l'existence d'applications externes précédemment lancées,
(b) un dispositif de recherche d'application (65) sensible aux demandes de recueil d'informations en provenance du dispositif de suivi d'application (62) qui renvoie au dispositif de suivi d'application (62) des informations pour savoir si une application externe est présente,
(c) un gestionnaire de groupe d'applications (61) qui crée, ajoute et enlève des objets d'entrée d'application, le gestionnaire de groupe d'applications (61) servant également à recevoir le nom de fichier externe en provenance de l'application de client (55) et à le sortir, et pour renvoyer un pointeur pour le fichier externe (97, 98, 99) et l'application externe à l'application de client, et
(d) un dispositif de lancement d'application (64) qui associe une application externe au nom de fichier externe sorti par ladite application de client (55) et qui lance l'application externe.

16. Procédé informatique selon la revendication 15, dans lequel ledit gestionnaire de fichiers (70) comprend :

(i) un dispositif de suivi de fichier (72) sensible à des demandes de recueil d'informations en provenance du dispositif de suivi d'application pour vérifier une modification du fichier externe créé par l'application de client renvoyant des informations concernant des modifications de fichier au dispositif de suivi d'application, et
(ii) un gestionnaire de groupe de fichiers (71) pour la création, l'ajout et l'enlèvement d'un objet d'entrée de fichier correspondant au fichier externe, le gestionnaire de groupe de fichiers (71) étant sensible à la sortie du nom de fichier externe en provenance du gestionnaire de groupe d'applications en créant un objet d'entrée de fichier.

17. Procédé selon l'une quelconque des revendications 13 à 16, comprenant de plus l'étape consistant à répondre (518) à la réception du pointeur pour le fichier externe et l'application externe en ordonnant à un système d'exploitation à fenêtres d'afficher une fenêtre pour l'application externe (S519).

18. Procédé selon l'une quelconque des revendications 13 à 16, dans lequel lesdites étapes consistant à répondre à la sélection par un utilisateur d'un élément dans le document composé comprennent les étapes consistant à déterminer si un pointeur valide pour l'élément sélectionné a été reçu en provenance du gestionnaire d'application (S506), ordonner (507) au système d'exploitation à fenêtres d'afficher une fenêtre pour l'application externe dans le cas où un pointeur valide existe, et envoyer (S509) un nom de fichier externe au gestionnaire d'application (60) dans un cas où un pointeur valide n'existe pas.

19. Procédé selon l'une quelconque des revendications 13 à 16, comprenant de plus l'étape consistant à

répondre pour fermer l'application externe en invalidant le pointeur envoyé à l'application de client (S714).

20. Procédé selon la revendication 19, comprenant de plus l'étape consistant à réimporter (S713) des données de fichier externe vers le document composé dans un cas où le fichier externe a été modifié.

21. Procédé selon l'une quelconque des revendications 13 à 16, comprenant de plus l'étape consistant à répondre pour fermer l'application de client en examinant des pointeurs (S801) reçus en provenance du gestionnaire d'application (60) pour déterminer si de quelconques applications externes sont toujours présentes, et à demander la fermeture d'applications externes toujours présentes (S805) avant la fermeture de l'application de client (55).

22. Procédé selon l'une quelconque des revendications 13 à 16, comprenant de plus les étapes consistant à répondre à la transformation en icône de l'application de client (55) en examinant (S801) les pointeurs reçus en provenance du gestionnaire d'application (60) pour déterminer si de quelconques applications externes sont toujours présentes et à ordonner au système d'exploitation à fenêtres de désafficher (S807) toute application externe toujours présente avant la transformation en icône.

23. Procédé selon la revendication 22, comprenant de plus l'étape consistant à répondre à la restauration de l'application de client transformée en icône en examinant des pointeurs reçus en provenance du gestionnaire d'application et à ordonner au système d'exploitation à fenêtres de réafficher (S811) toute application externe toujours présente.

24. Procédé informatique selon l'une quelconque des revendications 13 à 16, dans lequel des informations recueillies comprennent un identificateur de fenêtre pour l'application externe et un identificateur de tâche pour l'application externe, et dans lequel ledit gestionnaire d'application (60) détermine si l'application externe est présente en faisant correspondre l'identificateur de fenêtre et l'identificateur de tâche à des identificateurs de fenêtre et de tâche correspondants dans un objet d'entrée pour l'application externe.

25. Appareil pour gérer une application externe en réponse à la sélection par un utilisateur d'un élément dans un document composé affiché en provenance d'une application de client, ledit appareil comprenant :

un dispositif d'affichage (22) pour afficher des documents composés et des fenêtres correspondant à l'application de client et à des applications externes ;
une mémoire (25) pour le stockage d'étapes de programme correspondant à un système d'exploitation à fenêtres qui répond à des instructions pour afficher l'application de client et des applications externes sur ledit dispositif d'affichage (22), et pour le stockage de séquences d'instructions de programme correspondant à l'application de client ; et
un processeur (40) pour exécuter les séquences d'instructions de programme stockées dans ladite mémoire ;

caractérisé en ce que lesdites séquences d'instructions de programme stockées comprennent des étapes pour effectuer l'un quelconque des procédés des revendications 13 à 24.

26. Produit formant logiciel informatique portant des instructions pouvant être lues par une machine efficaces pour amener un appareil de traitement à exécuter un procédé selon l'une quelconque des revendications 13 à 24.

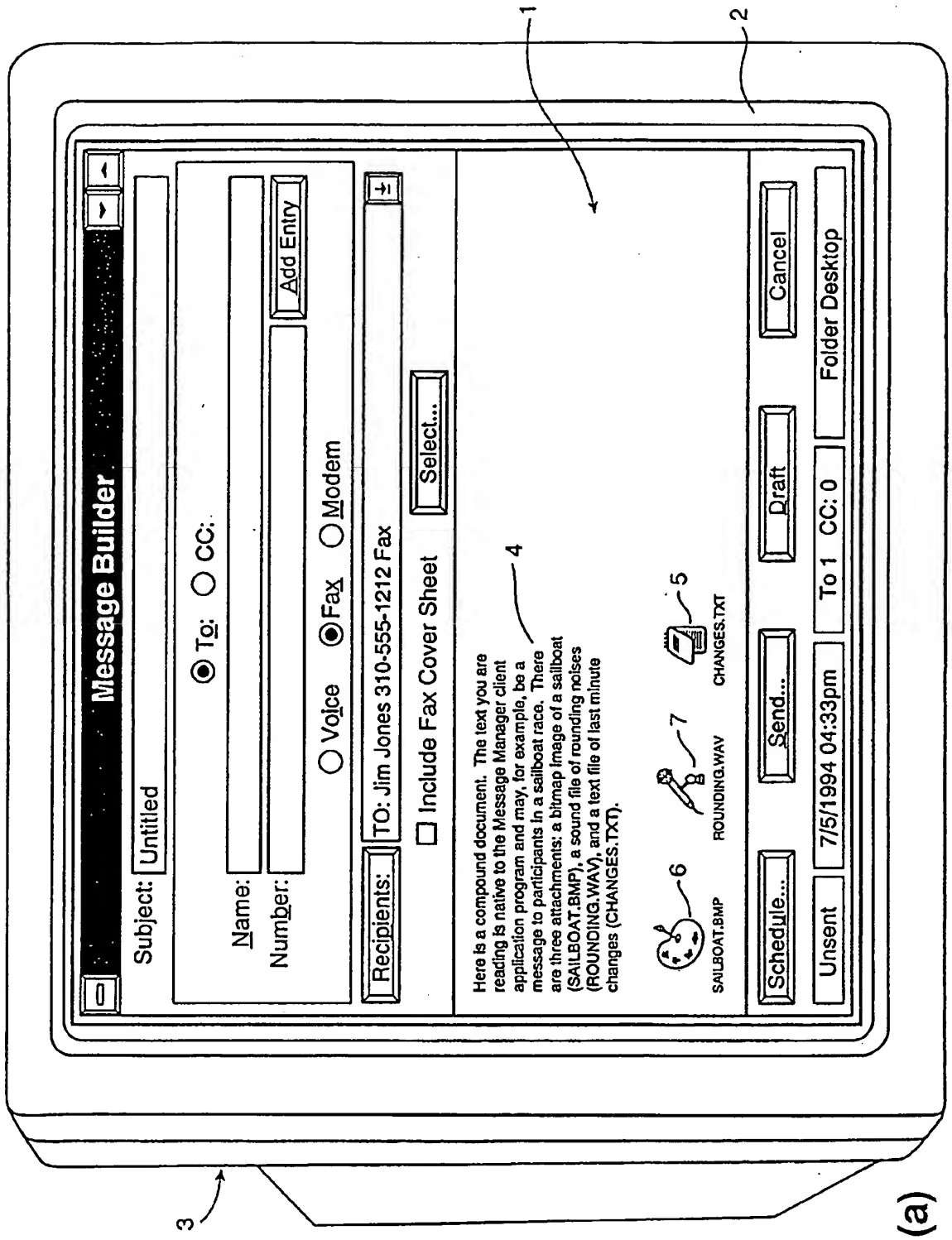
Message Builder
Subject: Untitled
To: CC:
Name:
Number:
Add Entry
Voice Fax Modem
Recipients:
TO: Jim Jones 310-555-1212 Fax
Include Fax Cover Sheet
Select...
Here is a compound document. The text you are reading is native to the Message Manager client application program and may, for example, be a message to participants in a sailboat race. There are three attachments: a bitmap image of a sailboat (SAILBOAT.BMP), a sound file of rounding noises (ROUNDING.WAV), and a text file of last minute changes (CHANGES.TXT).
SAILBOAT.BMP
ROUNDING.WAV
CHANGES.TXT
Schedule...
Send...
Draft
Cancel
Unsent
7/5/1994 04:33pm
To 1 CC: 0
Folder Desktop
1
2
3
4
5
6
7

FIG. 1(a)

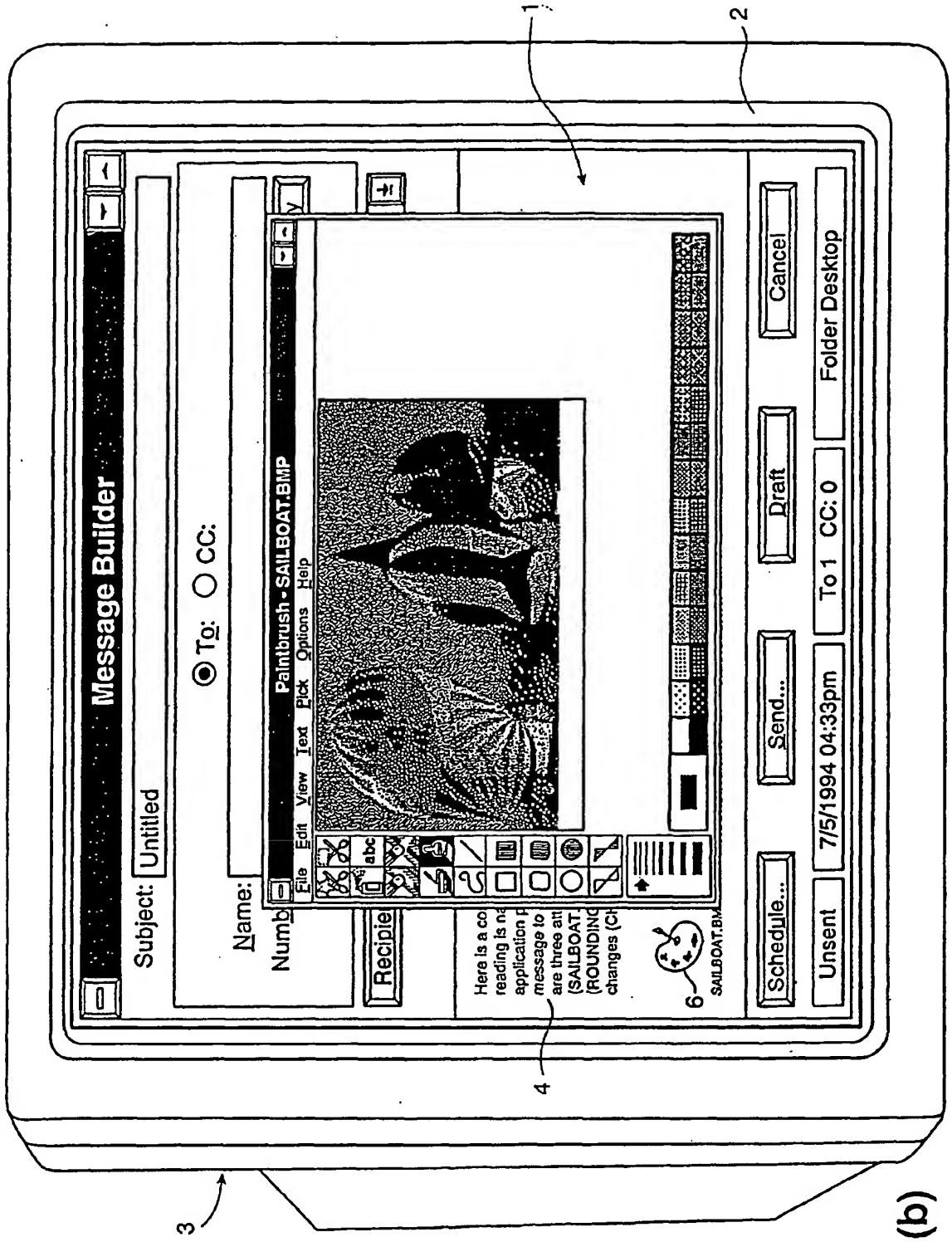
Message Builder
Subject: Untitled
To: CC:
Name:
Paintbrush - SAILBOAT.BMP
File Edit View Text Pick Options Help
Send...
Draft
Cancel
Schedule...
Unsent
7/5/1994 04:33pm
To 1 CC: 0
Folder Desktop
1
2
3
4
6

FIG. 1(b)

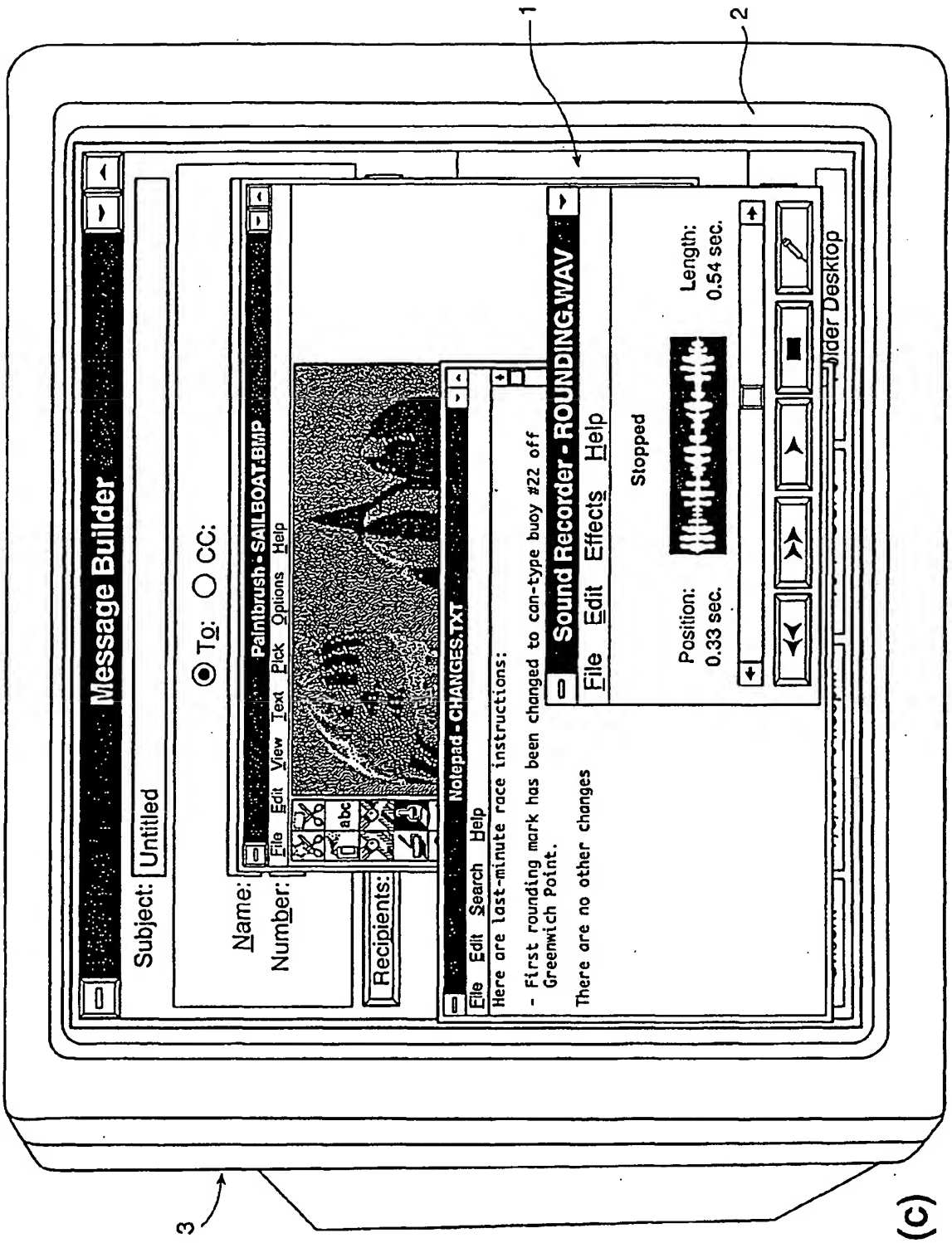
1
2
3
Message Builder
Subject: Untitled
To: CC:
Name:
Number:
Recipients:
Paintbrush - SAILBOAT.BMP
File Edit View Text Pick Options Help
Notepad - CHANGES.TXT
File Edit Search Help
Here are last-minute race instructions:
- First rounding mark has been changed to can-type buoy #22 off Greenwich Point.
There are no other changes
Sound Recorder - ROUNDING.WAV
File Edit Effects Help
Stopped
Position: 0.33 sec.
Length: 0.54 sec.
Desktop

FIG. 1(c)

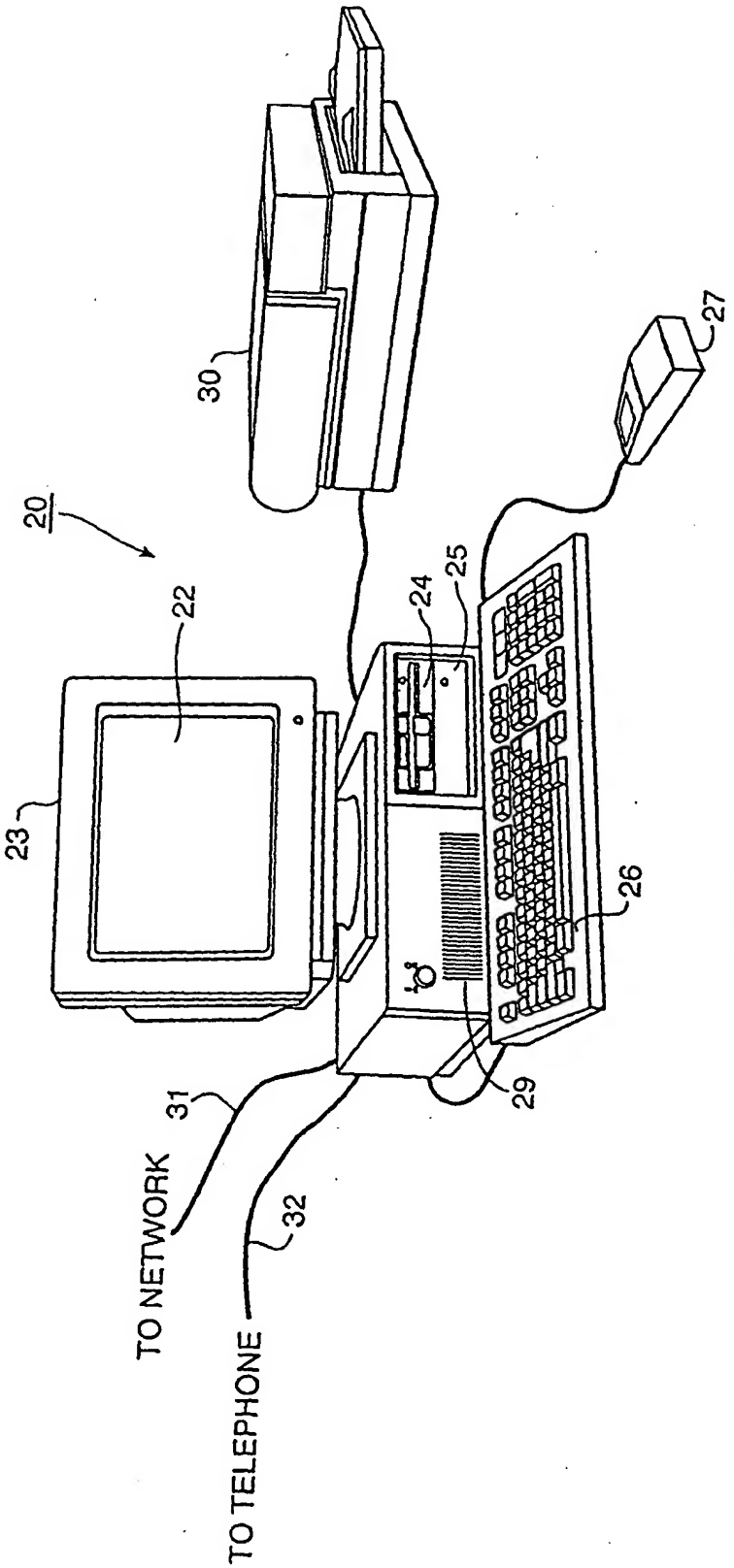

FIG. 2

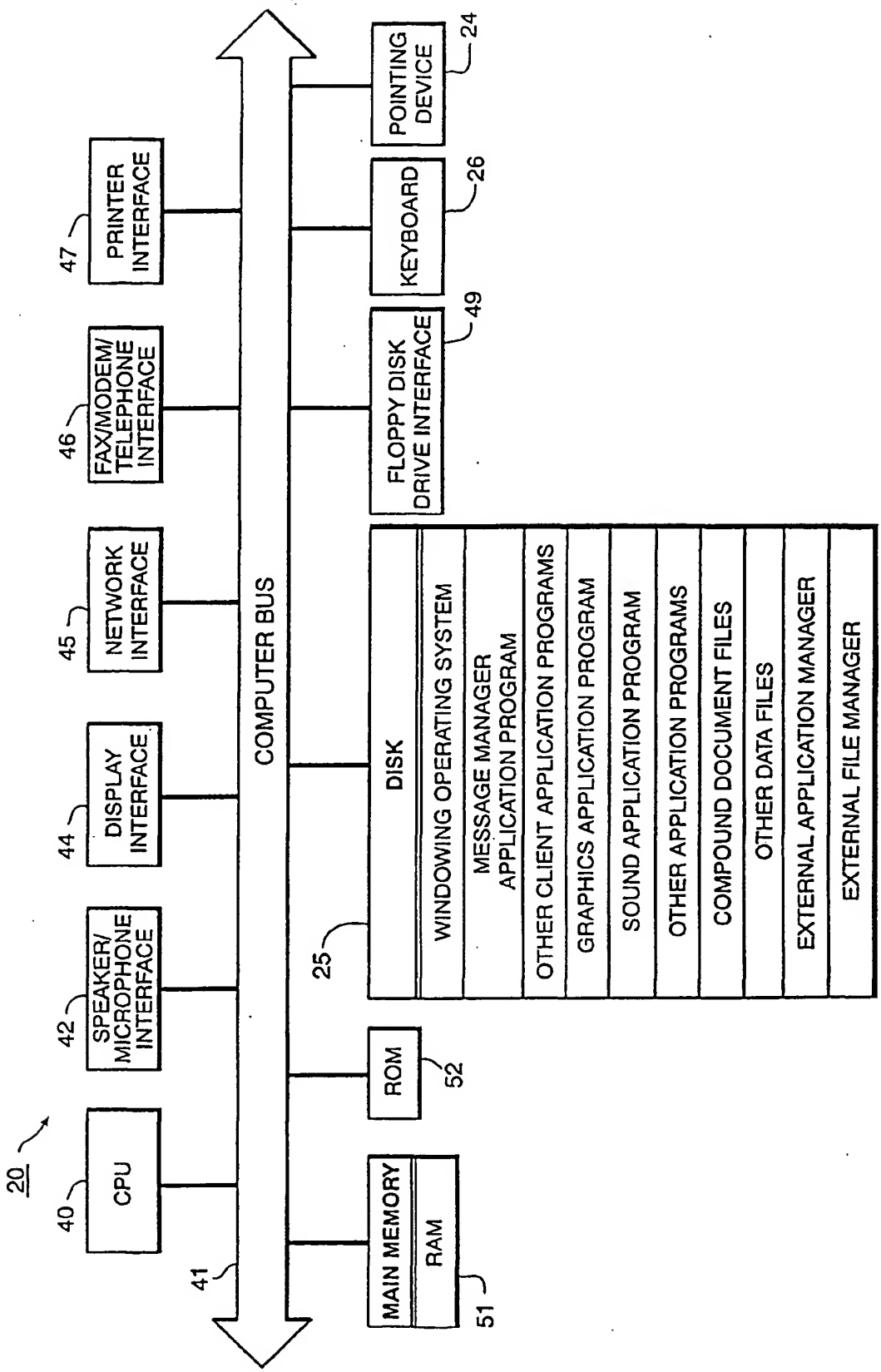
20
40
CPU
42
SPEAKER/ MICROPHONE INTERFACE
44
DISPLAY INTERFACE
45
NETWORK INTERFACE
46
FAX/MODEM/ TELEPHONE INTERFACE
47
PRINTER INTERFACE
41
COMPUTER BUS
MAIN MEMORY
RAM
51
ROM
52
25
DISK
WINDOWING OPERATING SYSTEM
MESSAGE MANAGER APPLICATION PROGRAM
OTHER CLIENT APPLICATION PROGRAMS
GRAPHICS APPLICATION PROGRAM
SOUND APPLICATION PROGRAM
OTHER APPLICATION PROGRAMS
COMPOUND DOCUMENT FILES
OTHER DATA FILES
EXTERNAL APPLICATION MANAGER
EXTERNAL FILE MANAGER
FLOPPY DISK DRIVE INTERFACE
49
KEYBOARD
26
POINTING DEVICE
24

FIG. 3

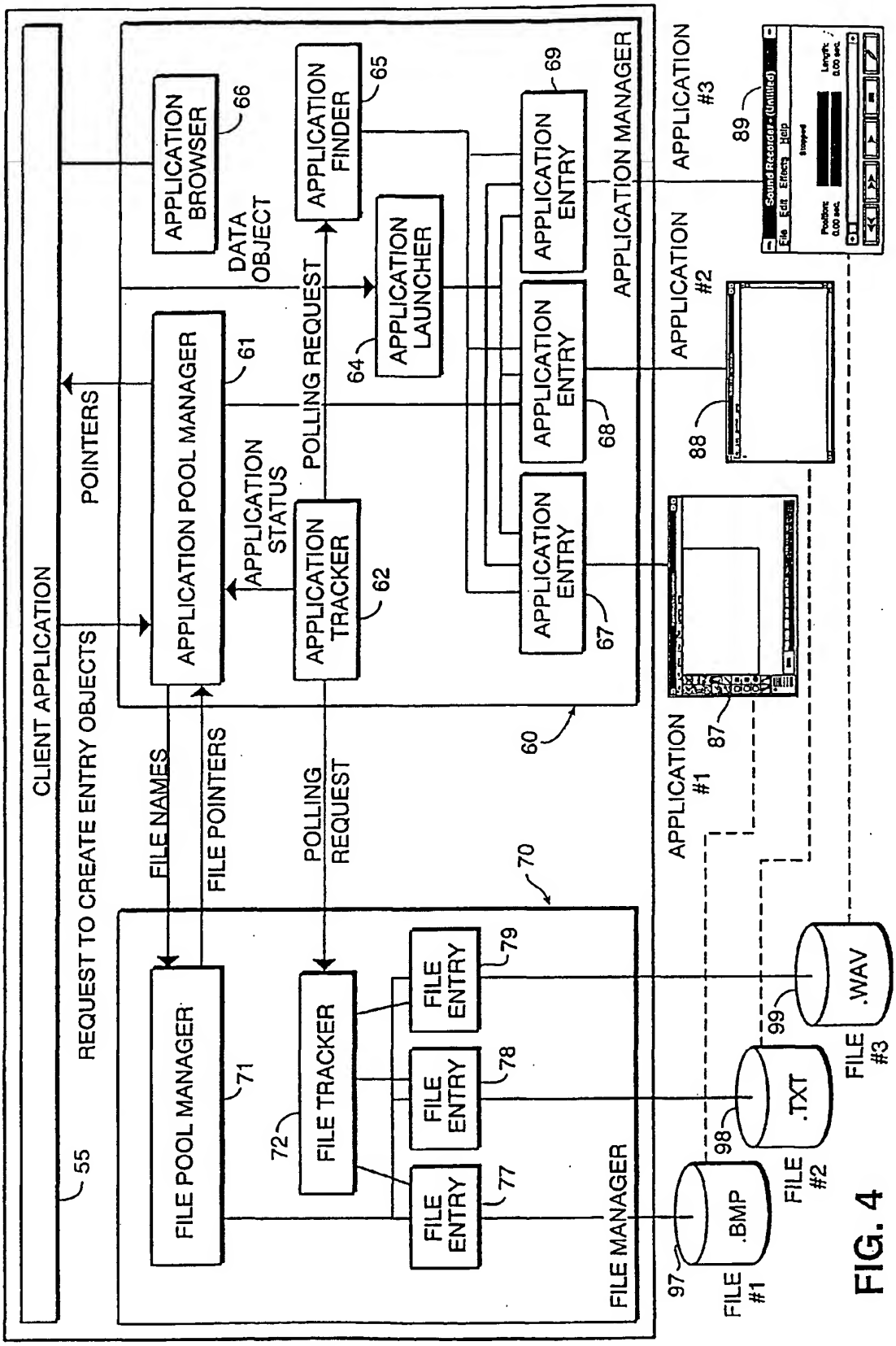
CLIENT APPLICATION
55
REQUEST TO CREATE ENTRY OBJECTS
POINTERS
APPLICATION BROWSER
66
DATA OBJECT
APPLICATION FINDER
65
APPLICATION POOL MANAGER
61
APPLICATION LAUNCHER
64
POLLING REQUEST
APPLICATION STATUS
APPLICATION TRACKER
62
APPLICATION ENTRY
69
APPLICATION ENTRY
68
APPLICATION ENTRY
67
APPLICATION MANAGER
60
APPLICATION #3
89
APPLICATION #2
88
APPLICATION #1
87
FILE NAMES
FILE POINTERS
POLLING REQUEST
70
FILE POOL MANAGER
71
FILE TRACKER
72
FILE ENTRY
79
FILE ENTRY
78
FILE ENTRY
77
FILE MANAGER
.WAV
99
FILE #3
.TXT
98
FILE #2
.BMP
97
FILE #1

FIG. 4

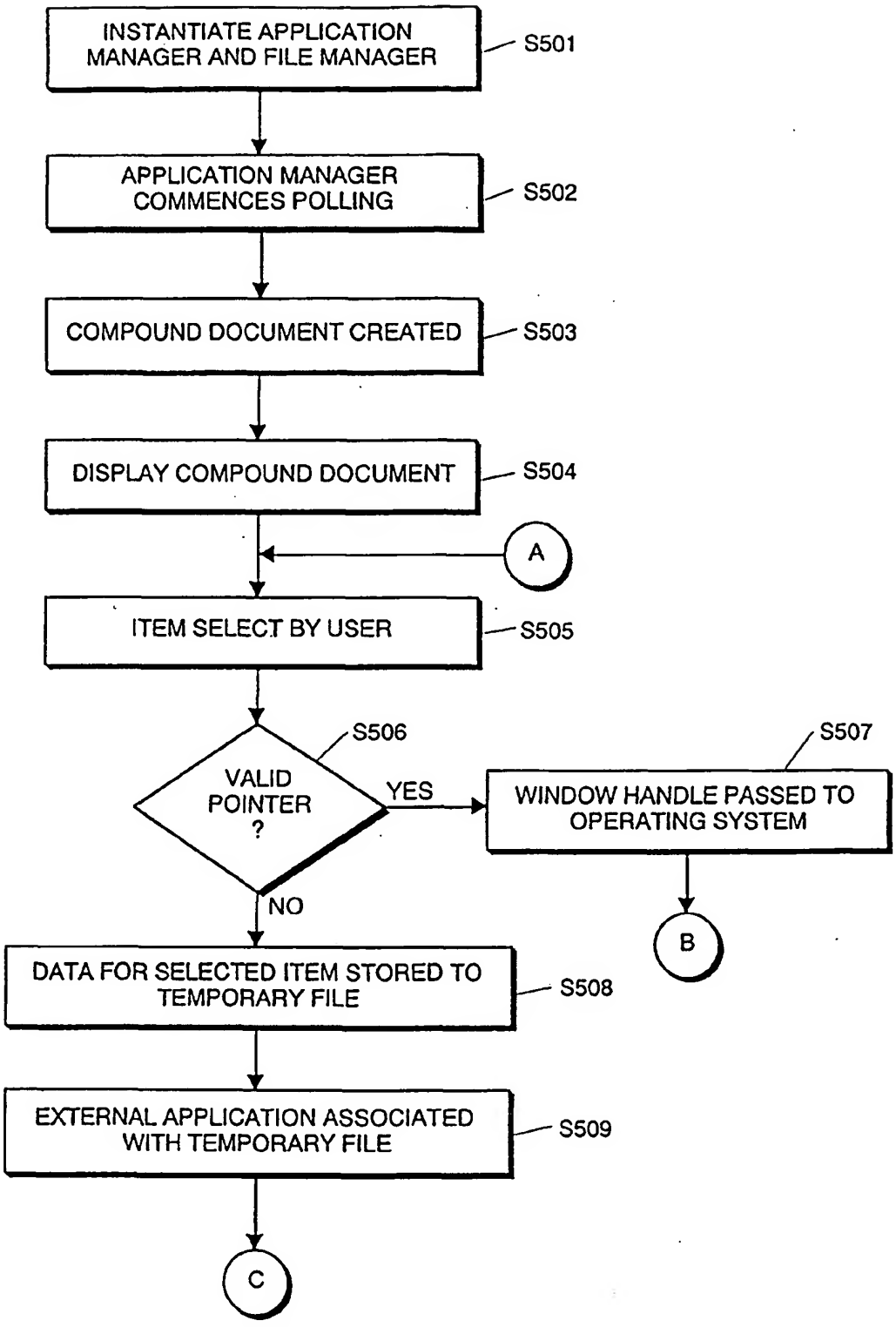
INSTANTIATE APPLICATION MANAGER AND FILE MANAGER
S501
APPLICATION MANAGER COMMENCES POLLING
S502
COMPOUND DOCUMENT CREATED
S503
DISPLAY COMPOUND DOCUMENT
S504
A
ITEM SELECT BY USER
S505
S506
VALID POINTER ?
YES
S507
WINDOW HANDLE PASSED TO OPERATING SYSTEM
B
NO
DATA FOR SELECTED ITEM STORED TO TEMPORARY FILE
S508
EXTERNAL APPLICATION ASSOCIATED WITH TEMPORARY FILE
S509
C

FIG. 5(a)

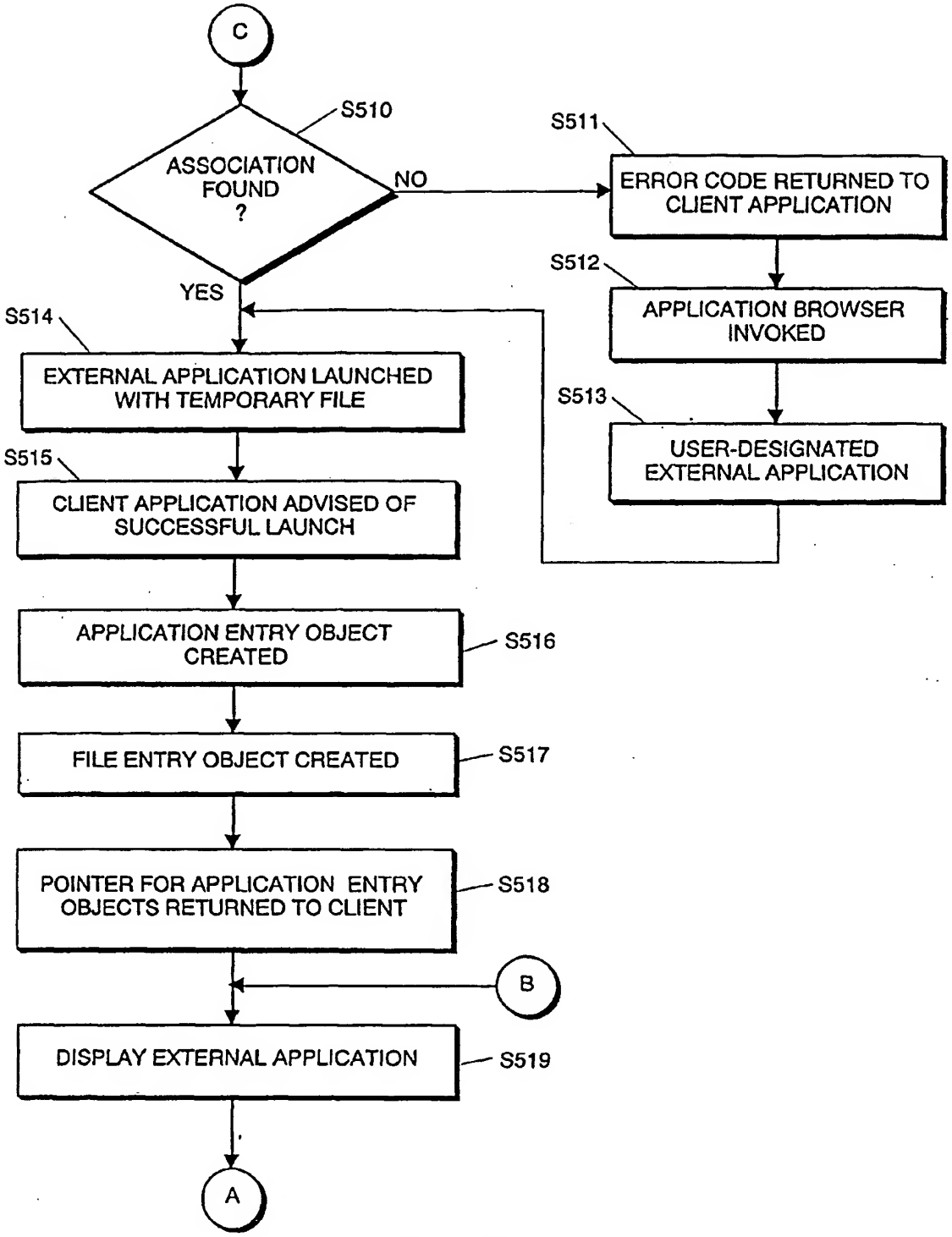
C
S510
ASSOCIATION FOUND ?
NO
S511
ERROR CODE RETURNED TO CLIENT APPLICATION
S512
APPLICATION BROWSER INVOKED
S513
USER-DESIGNATED EXTERNAL APPLICATION
YES
S514
EXTERNAL APPLICATION LAUNCHED WITH TEMPORARY FILE
S515
CLIENT APPLICATION ADVISED OF SUCCESSFUL LAUNCH
APPLICATION ENTRY OBJECT CREATED
S516
FILE ENTRY OBJECT CREATED
S517
POINTER FOR APPLICATION ENTRY OBJECTS RETURNED TO CLIENT
S518
B
DISPLAY EXTERNAL APPLICATION
S519
A

FIG. 5(b)

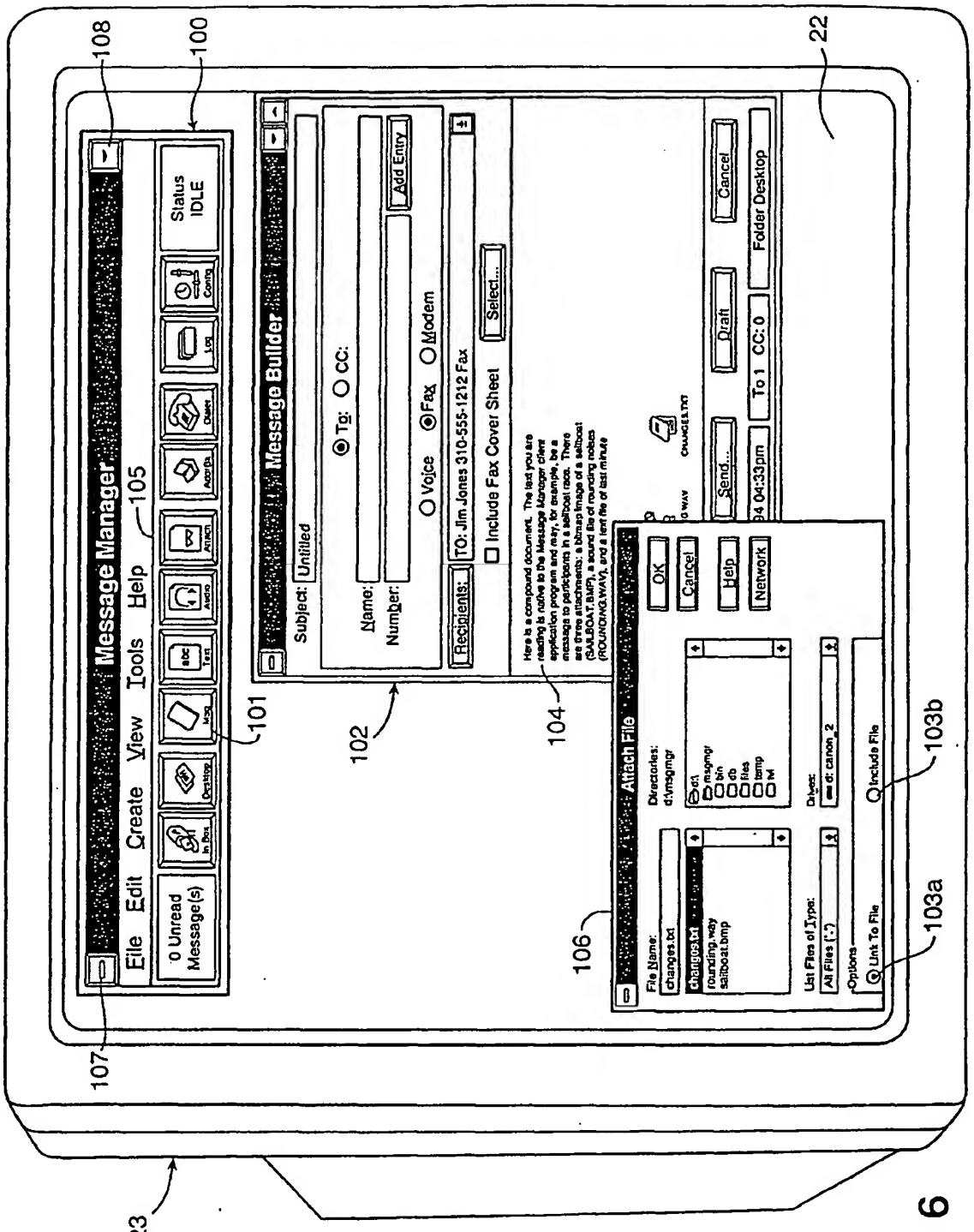
108
100
105
101
102
104
106
107
23
22
103b
103a
Message Manager
File Edit Create View Tools Help
0 Unread Message(s)
Status IDLE
Message Builder
Subject: Untitled
To: CC:
Name:
Number:
Voice Fax Modem
Add Entry
Recipients:
TO: Jim Jones 310-555-1212 Fax
Include Fax Cover Sheet
Select...
Send...
Draft
Cancel
To 1 CC: 0
Folder Desktop
Attach File
File Name:
changes.txt
Directories:
d:\msgmgr
OK
Cancel
Help
Network
Drives:
d: canon_2
List Files of Type:
All Files (*.*)
Options
Link To File
Include File

FIG. 6

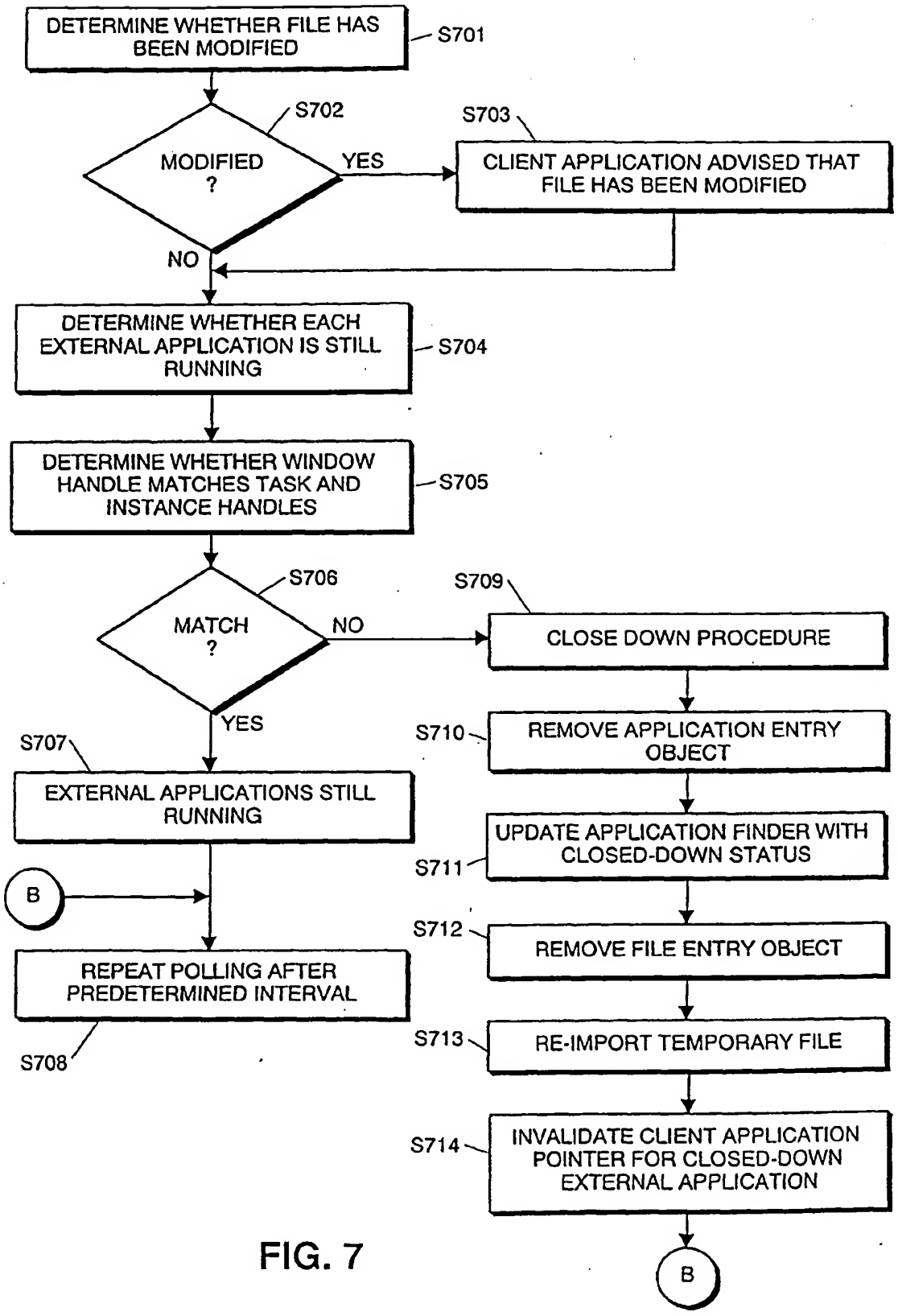
DETERMINE WHETHER FILE HAS BEEN MODIFIED
S701
S702
MODIFIED ?
YES
S703
CLIENT APPLICATION ADVISED THAT FILE HAS BEEN MODIFIED
NO
DETERMINE WHETHER EACH EXTERNAL APPLICATION IS STILL RUNNING
S704
DETERMINE WHETHER WINDOW HANDLE MATCHES TASK AND INSTANCE HANDLES
S705
S706
MATCH ?
NO
S709
CLOSE DOWN PROCEDURE
YES
S707
EXTERNAL APPLICATIONS STILL RUNNING
S710
REMOVE APPLICATION ENTRY OBJECT
UPDATE APPLICATION FINDER WITH CLOSED-DOWN STATUS
S711
B
S712
REMOVE FILE ENTRY OBJECT
REPEAT POLLING AFTER PREDETERMINED INTERVAL
S713
RE-IMPORT TEMPORARY FILE
S708
S714
INVALIDATE CLIENT APPLICATION POINTER FOR CLOSED-DOWN EXTERNAL APPLICATION
B

FIG. 7

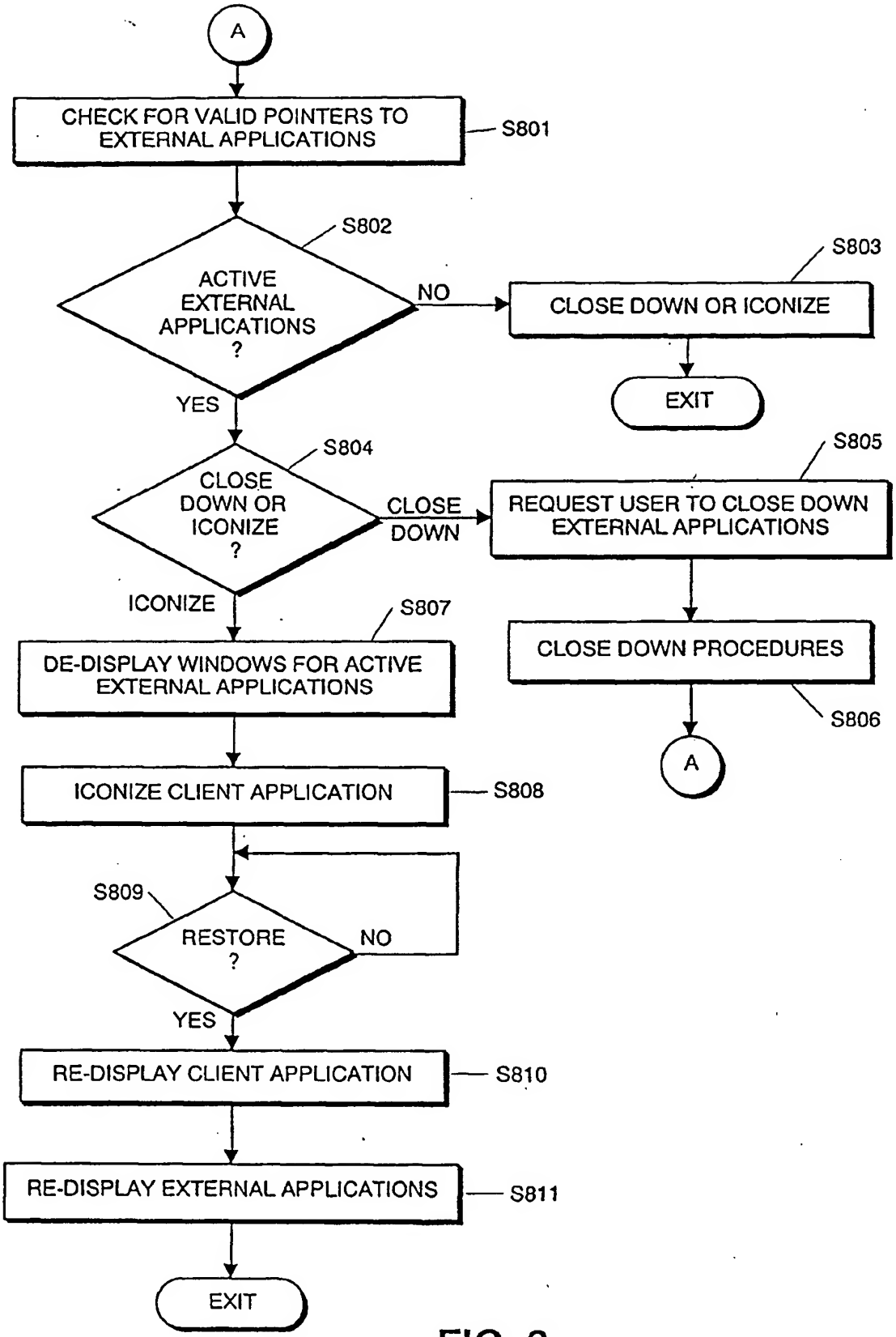
A
CHECK FOR VALID POINTERS TO EXTERNAL APPLICATIONS
S801
S802
ACTIVE EXTERNAL APPLICATIONS ?
NO
S803
CLOSE DOWN OR ICONIZE
EXIT
YES
S804
CLOSE DOWN OR ICONIZE ?
CLOSE DOWN
S805
REQUEST USER TO CLOSE DOWN EXTERNAL APPLICATIONS
ICONIZE
S807
DE-DISPLAY WINDOWS FOR ACTIVE EXTERNAL APPLICATIONS
CLOSE DOWN PROCEDURES
S806
A
ICONIZE CLIENT APPLICATION
S808
S809
RESTORE ?
NO
YES
RE-DISPLAY CLIENT APPLICATION
S810
RE-DISPLAY EXTERNAL APPLICATIONS
S811
EXIT

FIG. 8

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.